GM - repetition - Serial position analysis (v5)

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

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

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

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

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

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

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	558	35	129	NA	NA	722
2	67	NA	447	97	111	722
3	316	NA	173	217	16	722
4	310	NA	243	70	39	662
5	236	NA	218	74	39	567
6	212	1	139	73	22	447
7	180	NA	105	28	19	332
8	92	NA	55	26	4	177
9	76	NA	2	NA	6	84

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7728532	0.0484765	0.1786704	NA	NA	722
2	0.0927978	NA	0.6191136	0.1343490	0.1537396	722
3	0.4376731	NA	0.2396122	0.3005540	0.0221607	722
4	0.4682779	NA	0.3670695	0.1057402	0.0589124	662
5	0.4162257	NA	0.3844797	0.1305115	0.0687831	567
6	0.4742729	0.0022371	0.3109620	0.1633110	0.0492170	447

pos_factor	O	P	V	1	S	total
7	0.5421687	NA	0.3162651	0.0843373	0.0572289	332
8	0.5197740	NA	0.3107345	0.1468927	0.0225989	177
9	0.9047619	NA	0.0238095	NA	0.0714286	84

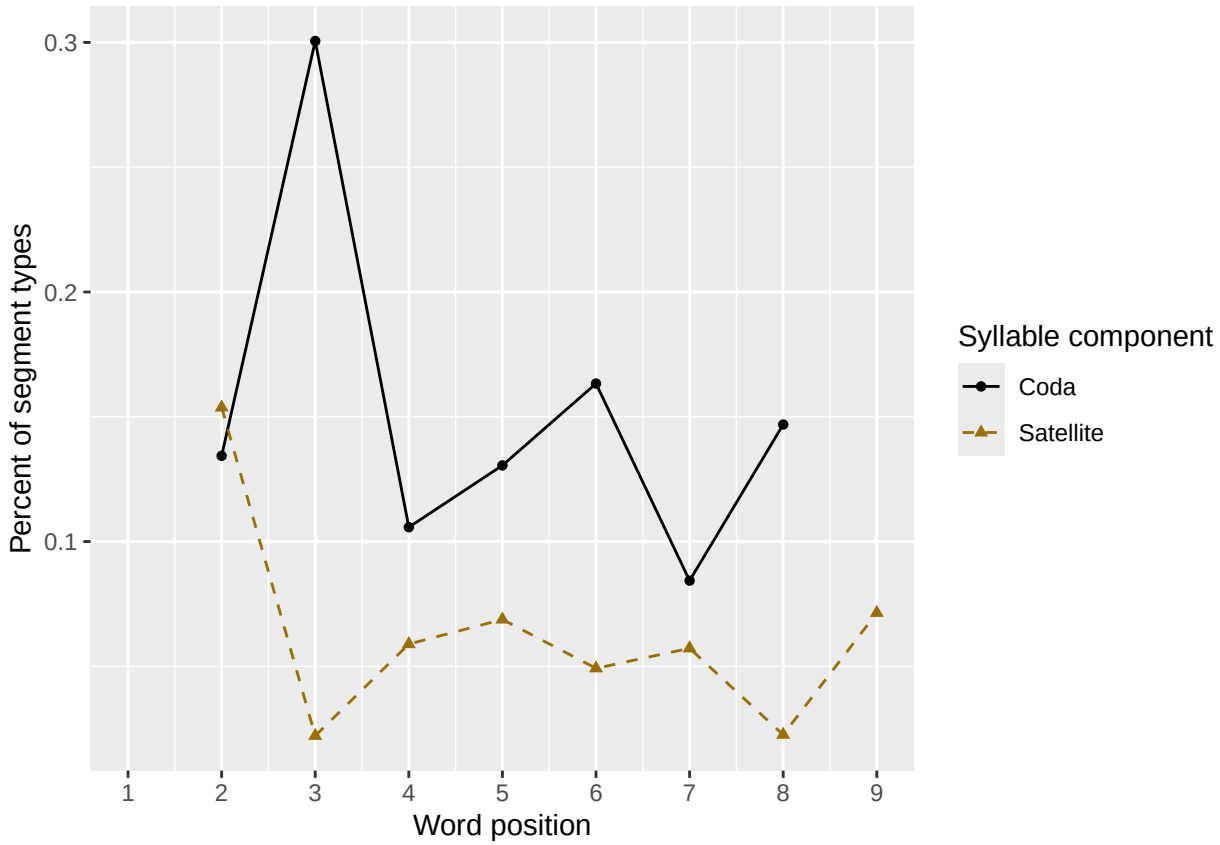
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.983 0.992 0.95  NA     NA     NA     NA     NA     NA
## 2       5 1     0.968 1     0.958 NA     NA     NA     NA     NA
## 3       6 1     0.975 1     0.967  0.917 NA     NA     NA     NA
## 4       7 1     0.991 0.990 0.952  0.968  0.959 NA     NA     NA
## 5       8 0.974 0.987 0.973 0.937  0.894  0.935  0.897 NA     NA
## 6       9 0.982 0.966 0.944 0.962  0.962  0.957  0.968  0.968 NA
## 7      10 0.988 0.976 0.937 0.946  0.929  0.945  0.921  0.940  0.917
```

```
# 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.
```

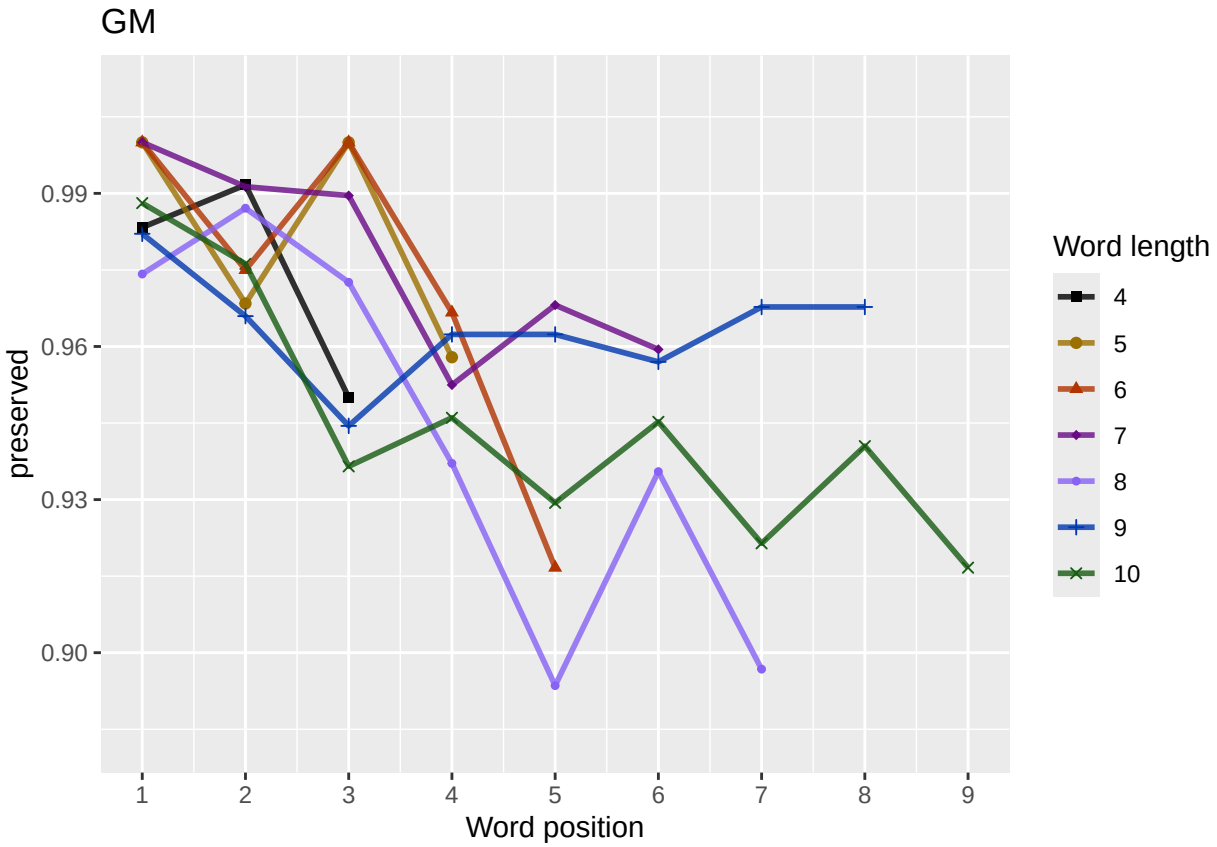
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    60    60    60    NA    NA    NA    NA    NA    NA
## 2       5    95    95    95    95    NA    NA    NA    NA    NA
## 3       6   120   120   120   120   120    NA    NA    NA    NA
## 4       7   115   115   115   115   115   115    NA    NA    NA
## 5       8   155   155   155   155   155   155   155    NA    NA
## 6       9    93    93    93    93    93    93    93    93    NA
## 7      10    84    84    84    84    84    84    84    84    84
```

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

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

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



Length and position

```
# length and position

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

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

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

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

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.88386     -0.08883      0.05972     -0.77676
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1375  AIC: 1505
## log likelihood:  -687.6337
## Nagelkerke R2:  0.04663445
## % pres/err predicted correctly:  -331.5168
## % of predictable range [ (model-null)/(1-null) ]:  0.01237075
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         8.462092         -0.427024          0.126988         -1.754283         -0.009421
##      stimlen:pos
##         0.130241
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4429 Residual
## Null Deviance:       1433
## Residual Deviance: 1373  AIC: 1507
## log likelihood:  -686.3058
## Nagelkerke R2:  0.048775
## % pres/err predicted correctly:  -331.3873
## % of predictable range [ (model-null)/(1-null) ]:  0.01275563
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      5.2364       0.0553      -0.7653
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1378  AIC: 1507
## log likelihood:  -688.8618
## Nagelkerke R2:  0.04465363
## % pres/err predicted correctly:  -331.7343
## % of predictable range [ (model-null)/(1-null) ]:  0.01172487
## **************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     7.43035     -0.39590     -0.96955      0.08718
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1377  AIC: 1509
## log likelihood:  -688.6071
## Nagelkerke R2:  0.04506443
## % pres/err predicted correctly:  -331.7038
## % of predictable range [ (model-null)/(1-null) ]:  0.01181559
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      4.1820      -0.2297
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1389  AIC: 1519
## log likelihood:  -694.4637
## Nagelkerke R2:  0.03560404
## % pres/err predicted correctly:  -332.4857
## % of predictable range [ (model-null)/(1-null) ]:  0.009493139
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.51845     -0.05315     -0.21082
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1388  AIC: 1519
## log likelihood:  -694.0105
## Nagelkerke R2:  0.0363371
## % pres/err predicted correctly:  -332.3991
## % of predictable range [ (model-null)/(1-null) ]:  0.009750379
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      4.6004      -0.1792
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1418  AIC: 1546
## log likelihood:  -709.2487
## Nagelkerke R2:  0.01160977
## % pres/err predicted correctly:  -334.4819
## % of predictable range [ (model-null)/(1-null) ]:  0.003563977
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.186
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4434 Residual
## Null Deviance:       1433
## Residual Deviance: 1433  AIC: 1559
## log likelihood:  -716.3673
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -335.6818
## % 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	1505.429	0.000000	1.000000	0.4640187	0.0466345	5.883857	-0.0888251	-1.7767619	NA	0.0597162	NA
preserved ~ stimlen * (I(pos^2) + pos)	1506.765	1.335769	0.512792	0.2379426	0.0487750	5.462092	-0.4270239	-1.7542830	0.1302408	0.1269881	-0.0094209

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	1506.88	0.4565877	0.4827320	0.2223994	0.2044653	6.236382	NA	-0.7653311	NA	0.0552979	NA
preserved ~ stimlen * pos	1509.12	2.6999783	0.1572393	0.0729610	0.2045067	4.430349	-0.3959034	-1.9695495	0.0871782	NA	NA
preserved ~ pos	1518.64	12.2122980	0.0013520	0.0006274	0.2035604	4.181985	NA	-0.2297194	NA	NA	NA
preserved ~ stimlen + pos	1519.25	13.8286420	0.0009985	0.0004610	0.2036337	4.518450	-0.0531501	-1.2108225	NA	NA	NA
preserved ~ stimlen	1545.79	40.3699260	0.0000000	0.0000000	0.0116048	3.600376	-0.1792198	NA	NA	NA	NA
preserved ~ 1	1559.22	53.7964740	0.0000000	0.0000000	0.0000000	3.186306	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
##     5.88386     -0.08883      0.05972     -0.77676
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1375  AIC: 1505
```

```
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
```

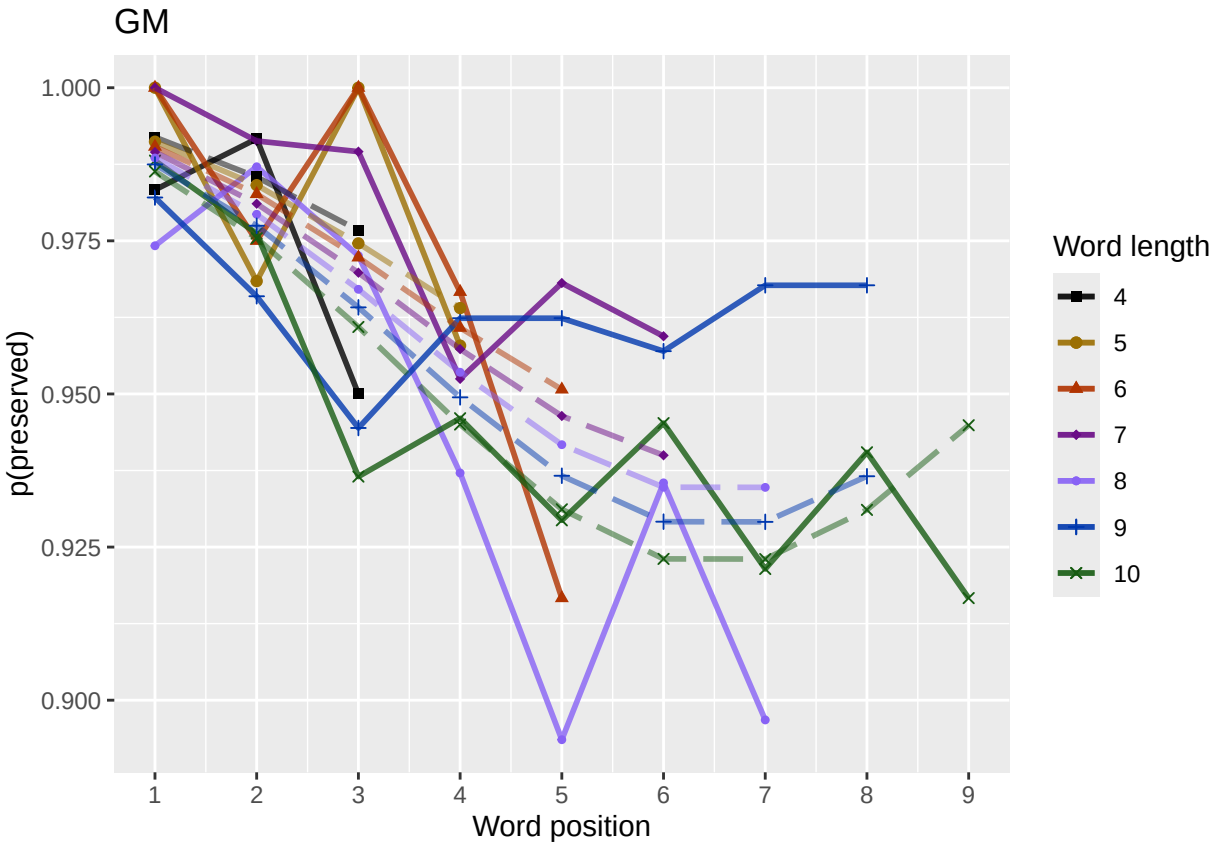
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.992 0.985 0.977 NA     NA     NA     NA     NA     NA
## 2       5 0.991 0.984 0.975  0.964 NA     NA     NA     NA     NA
## 3       6 0.990 0.983 0.972  0.961  0.951 NA     NA     NA     NA
## 4       7 0.989 0.981 0.970  0.957  0.946  0.940 NA     NA     NA
## 5       8 0.989 0.979 0.967  0.954  0.942  0.935  0.935 NA     NA
## 6       9 0.987 0.977 0.964  0.949  0.937  0.929  0.929  0.937 NA
```

```
## 7      10 0.986 0.975 0.961  0.945  0.931  0.923  0.923  0.931  0.945
```

```
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
```

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1        7   722
```

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

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

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

```
## [1] "The number of responses with fragments was 7 / 722 = 0.97 percent"
```

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

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

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

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

```
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
```

```
##     5.97346     -0.08821      0.07359     -0.84837
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4411 Residual
## Null Deviance:       1301
## Residual Deviance: 1259  AIC: 1391
## log likelihood:  -629.6632
## Nagelkerke R2:  0.03714138
## % pres/err predicted correctly:  -295.936
## % of predictable range [ (model-null)/(1-null) ]:  0.009468476
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     5.32890      0.06908     -0.83603
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4412 Residual
## Null Deviance:       1301
## Residual Deviance: 1262  AIC: 1392
## log likelihood:  -630.812
## Nagelkerke R2:  0.0351217
## % pres/err predicted correctly:  -296.143
## % of predictable range [ (model-null)/(1-null) ]:  0.008777941
## **************************
## 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)
##         7.889283         -0.356915          0.078712         -1.386936         -0.002844
##      stimlen:pos
##         0.084069
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4409 Residual
## Null Deviance:       1301
## Residual Deviance: 1257  AIC: 1392
## log likelihood:  -628.2537
## Nagelkerke R2:  0.03961787
## % pres/err predicted correctly:  -295.7793
## % of predictable range [ (model-null)/(1-null) ]:  0.009991241
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##      7.7011      -0.4391      -1.0580       0.1029
```

```
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4411 Residual
## Null Deviance:       1301
## Residual Deviance: 1262  AIC: 1396
## log likelihood:  -631.1542
## Nagelkerke R2:  0.03451987
## % pres/err predicted correctly:  -296.2475
## % of predictable range [ (model-null)/(1-null) ]:  0.008429321
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      4.0891      -0.1852
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4413 Residual
## Null Deviance:       1301
## Residual Deviance: 1276  AIC: 1408
## log likelihood:  -638.1201
## Nagelkerke R2:  0.0222489
## % pres/err predicted correctly:  -297.2244
## % of predictable range [ (model-null)/(1-null) ]:  0.005170682
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.39484     -0.04798     -0.16856
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4412 Residual
## Null Deviance:       1301
## Residual Deviance: 1276  AIC: 1409
## log likelihood:  -637.7719
## Nagelkerke R2:  0.02286328
## % pres/err predicted correctly:  -297.1643
## % of predictable range [ (model-null)/(1-null) ]:  0.005371106
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.4438      -0.1449
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4413 Residual
## Null Deviance:       1301
```

```
## Residual Deviance: 1293  AIC: 1423
## log likelihood:  -646.469
## Nagelkerke R2:  0.007490643
## % pres/err predicted correctly:  -298.1021
## % of predictable range [ (model-null)/(1-null) ]:  0.002242547
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.307
##
## Degrees of Freedom: 4414 Total (i.e. Null);  4414 Residual
## Null Deviance:       1301
## Residual Deviance: 1301  AIC: 1431
## log likelihood:  -650.6945
## Nagelkerke R2:  8.697615e-16
## % pres/err predicted correctly:  -298.7744
## % 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	1390.965	0.000000	1.000000	0.4729828	0.0371454	3.973464	-0.0882143	-3.8483718	NA	0.0735893	NA
preserved ~ I(pos^2) + pos	1392.263	1.298188	0.5225188	0.2471424	0.0351257	3.328899	NA	-0.8360259	NA	0.0690819	NA
preserved ~ stimlen * (I(pos^2) + pos)	1392.265	1.300670	0.5218710	0.2468360	0.0396179	3.889284	-0.3569152	-3.8693623	0.0840695	0.0787117	-0.002844
preserved ~ stimlen * pos	1396.295	5.331626	0.0695428	0.0328925	0.0345197	3.701071	-0.4390769	-3.9058038	0.1028910	NA	NA
preserved ~ pos	1408.124	17.159205	0.0001879	0.0000889	0.0222489	3.089100	NA	-0.1851538	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	1408.998	18.033766	0.0001210	0.0000574	0.0228643	4.394836	-0.0479810	-0.1685610	NA	NA	NA
preserved ~ stimlen	1423.203	32.240258	0.0000001	0.0000000	0.0074946	4.443750	-0.1449132	NA	NA	NA	NA
preserved ~ 1	1430.773	39.810597	0.0000000	0.0000000	0.0000000	3.306934	NA	NA	NA	NA	NA

```r
# plot no fragment data

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

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

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

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

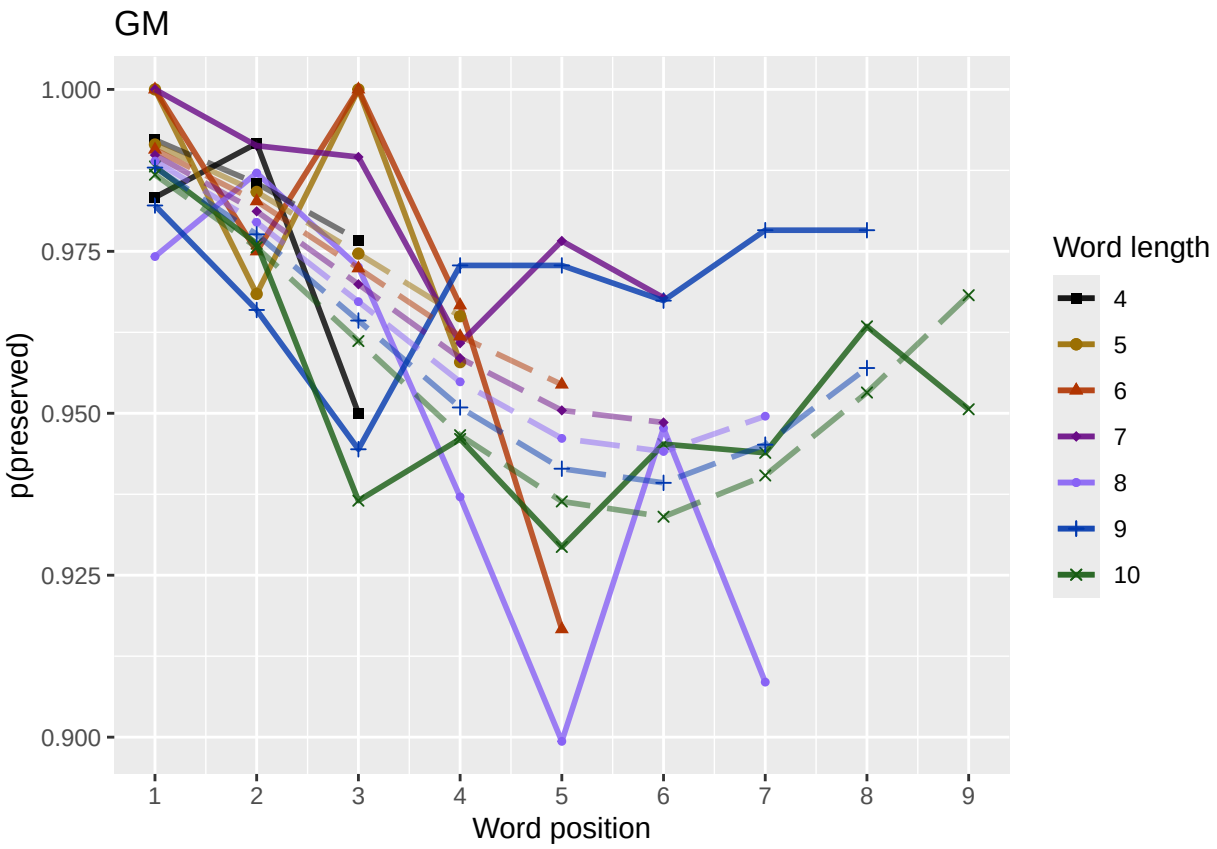
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.992 0.985 0.977 NA     NA     NA     NA     NA     NA
## 2       5 0.991 0.984 0.975  0.965 NA     NA     NA     NA     NA
## 3       6 0.991 0.983 0.972  0.962  0.954 NA     NA     NA     NA
## 4       7 0.990 0.981 0.970  0.959  0.950  0.949 NA     NA     NA
## 5       8 0.989 0.979 0.967  0.955  0.946  0.944  0.950 NA     NA
## 6       9 0.988 0.978 0.964  0.951  0.941  0.939  0.945  0.957 NA
## 7      10 0.987 0.976 0.961  0.947  0.936  0.934  0.940  0.953  0.968
```

```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: 0.88 - 1.01"
```

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

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

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

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

```r
# if there is an effect of length and we just average position data,
# later positions) will have a lower average (because of the length effect)
# even if all position functions go upward

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.00284567
```

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

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

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

```r
print(OA_mean_pos_diff)
```

```
## [1] -0.00870727
```

```r
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] 0.01077856
```

```
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] 0.01523031 0.02716729 0.03964225 0.04950545 0.05377910 0.05835832 0.06328836
## [1] "differences from min to right max for each row: "
## [1] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.007447014 0.021852684
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.06328836
## [1] 0.02185268
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.3452876
```

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

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

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

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

```
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     5.59913     -0.04965      0.05919     -0.77181      0.12594
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4430 Residual
## Null Deviance:       1433
## Residual Deviance: 1366  AIC: 1497
## log likelihood:  -683.1536
## Nagelkerke R2:  0.05385135
## % pres/err predicted correctly:  -330.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.01588845
## ***************************
```

```
## 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
##           5.41442            0.06956           -0.87011            0.36019            0.01488
##      pos:log_freq
##          -0.13019
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4429 Residual
## Null Deviance:       1433
## Residual Deviance: 1364  AIC: 1497
## log likelihood:  -682.0304
## Nagelkerke R2:  0.05565836
## % pres/err predicted correctly:  -330.1836
## % of predictable range [ (model-null)/(1-null) ]:  0.01633078
## **************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           5.83822           -0.05768            0.07270           -0.88004            0.34009
## I(pos^2):log_freq       pos:log_freq
##           0.01507           -0.12900
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4428 Residual
## Null Deviance:       1433
## Residual Deviance: 1363  AIC: 1497
## log likelihood:  -681.5543
## Nagelkerke R2:  0.05642395
## % pres/err predicted correctly:  -330.1079
## % of predictable range [ (model-null)/(1-null) ]:  0.0165555
## **************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         5.600200         -0.049409          0.110213          0.059273         -0.772590
## stimlen:log_freq
##         0.001936
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4429 Residual
## Null Deviance:       1433
## Residual Deviance: 1366  AIC: 1499
## log likelihood:  -683.151
## Nagelkerke R2:  0.05385553
```

```
## % pres/err predicted correctly:  -330.3259
## % of predictable range [ (model-null)/(1-null) ]:  0.01590811
## **************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##           5.84562           -0.06106            0.46599            0.07315           -0.88017
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##          -0.01654            0.01599           -0.13250
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4427 Residual
## Null Deviance:       1433
## Residual Deviance: 1363  AIC: 1499
## log likelihood:  -681.4034
## Nagelkerke R2:  0.05666662
## % pres/err predicted correctly:  -330.1039
## % of predictable range [ (model-null)/(1-null) ]:  0.01656755
## **************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.88386     -0.08883      0.05972     -0.77676
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1375  AIC: 1505
## log likelihood:  -687.6337
## Nagelkerke R2:  0.04663445
## % pres/err predicted correctly:  -331.5168
## % of predictable range [ (model-null)/(1-null) ]:  0.01237075
## **************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         8.462092         -0.427024          0.126988         -1.754283         -0.009421
##      stimlen:pos
##         0.130241
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4429 Residual
## Null Deviance:       1433
## Residual Deviance: 1373  AIC: 1507
## log likelihood:  -686.3058
```

```
## Nagelkerke R2:  0.048775
## % pres/err predicted correctly:  -331.3873
## % of predictable range [ (model-null)/(1-null) ]:  0.01275563
## **************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      5.2364       0.0553      -0.7653
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1378  AIC: 1507
## log likelihood:  -688.8618
## Nagelkerke R2:  0.04465363
## % pres/err predicted correctly:  -331.7343
## % of predictable range [ (model-null)/(1-null) ]:  0.01172487
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      4.1563      -0.2159       0.1301
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1379  AIC: 1508
## log likelihood:  -689.4089
## Nagelkerke R2:  0.04377074
## % pres/err predicted correctly:  -331.1596
## % of predictable range [ (model-null)/(1-null) ]:  0.01343196
## **************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     7.43035     -0.39590     -0.96955      0.08718
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1377  AIC: 1509
## log likelihood:  -688.6071
## Nagelkerke R2:  0.04506443
## % pres/err predicted correctly:  -331.7038
## % of predictable range [ (model-null)/(1-null) ]:  0.01181559
```

```
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##      4.2430      -0.0135      -0.2115       0.1279
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1379  AIC: 1510
## log likelihood:  -689.3816
## Nagelkerke R2:  0.04381484
## % pres/err predicted correctly:  -331.1604
## % of predictable range [ (model-null)/(1-null) ]:  0.01342949
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     4.147607     -0.213617      0.111623      0.003874
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1379  AIC: 1510
## log likelihood:  -689.3863
## Nagelkerke R2:  0.04380723
## % pres/err predicted correctly:  -331.1183
## % of predictable range [ (model-null)/(1-null) ]:  0.01355449
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     4.244660     -0.015321     -0.208278      0.106133      0.004485
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4430 Residual
## Null Deviance:       1433
## Residual Deviance: 1379  AIC: 1512
## log likelihood:  -689.3517
## Nagelkerke R2:  0.04386312
## % pres/err predicted correctly:  -331.1126
## % of predictable range [ (model-null)/(1-null) ]:  0.01357135
## **************************
## model index:  6
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         4.244159         -0.014034          0.162208         -0.211451         -0.004231
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4430 Residual
## Null Deviance:       1433
## Residual Deviance: 1379  AIC: 1512
## log likelihood:  -689.3682
## Nagelkerke R2:  0.04383637
## % pres/err predicted correctly:  -331.1799
## % of predictable range [ (model-null)/(1-null) ]:  0.0133714
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         4.247685         -0.017686          0.168104         -0.206047         -0.009642
##     log_freq:pos
##         0.007838
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4429 Residual
## Null Deviance:       1433
## Residual Deviance: 1379  AIC: 1514
## log likelihood:  -689.299
## Nagelkerke R2:  0.04394816
## % pres/err predicted correctly:  -331.1206
## % of predictable range [ (model-null)/(1-null) ]:  0.01354756
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      4.1820      -0.2297
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1389  AIC: 1519
## log likelihood:  -694.4637
## Nagelkerke R2:  0.03560404
## % pres/err predicted correctly:  -332.4857
## % of predictable range [ (model-null)/(1-null) ]:  0.009493139
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.51845     -0.05315     -0.21082
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1388  AIC: 1519
## log likelihood:  -694.0105
## Nagelkerke R2:  0.0363371
## % pres/err predicted correctly:  -332.3991
## % of predictable range [ (model-null)/(1-null) ]:  0.009750379
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      4.3272      -0.1403       0.1267
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1409  AIC: 1537
## log likelihood:  -704.6652
## Nagelkerke R2:  0.01906543
## % pres/err predicted correctly:  -333.4772
## % of predictable range [ (model-null)/(1-null) ]:  0.006548269
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         4.328489         -0.140802          0.161284         -0.004257
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1409  AIC: 1539
## log likelihood:  -704.6516
## Nagelkerke R2:  0.01908747
## % pres/err predicted correctly:  -333.4934
## % of predictable range [ (model-null)/(1-null) ]:  0.006500062
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)      stimlen
##      4.6004      -0.1792
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1418  AIC: 1546
## log likelihood:  -709.2487
## Nagelkerke R2:  0.01160977
## % pres/err predicted correctly:  -334.4819
## % of predictable range [ (model-null)/(1-null) ]:  0.003563977
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.186
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4434 Residual
## Null Deviance:       1433
## Residual Deviance: 1433  AIC: 1559
## log likelihood:  -716.3673
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -335.6818
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	log_freq	stimlen	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos + log_freq	1496.577	0.000000	1.0000000	[illegible]	[illegible]	[illegible]	-0.0496536	0.1259359	NA	-0.7718053	NA	NA	0.0591947	NA	NA	NA

Model	AIC	Delta_AIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ (I(pos^2) + pos) * log_freq	1496.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.3601931	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	1497.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0576804	[illegible]	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	1498.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0494095	[illegible]	[illegible]	-0.7725899	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	1499.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0610634	0.4659915	[illegible]	[illegible]	NA	-0.1325001	[illegible]	NA	[illegible]	NA	NA
preserved ~ stimlen + I(pos^2) + pos	1505.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0888251	NA	NA	-0.7767619	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1506.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.4270239	NA	NA	-1.7542830	NA	NA	[illegible]	NA	NA	0.1302408	-0.0094209
preserved ~ I(pos^2) + pos	1506.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	-0.7653311	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ pos + log_freq	1508.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.2158559	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	1509.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3959034	NA	NA	-0.9695495	NA	NA	NA	NA	NA	[illegible]	NA
preserved ~ stimlen + pos + log_freq	1510.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0134991	[illegible]	NA	-0.2114548	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	1510.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.2136170	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	1511.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0153205	[illegible]	NA	-0.2082779	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	1512.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0140343	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAIC AICexp AICw NagR2 (Intercept) stimlen log_freq stimlen:log_freq pos log_freq:pos I(pos^2) log_freq:I(pos^2) stimlen:pos stimlen:I(pos^2)										
preserved ~ stimlen * log_freq + pos * log_freq	1513.[illegible]	-0.0176858	0.1681042	-0.0096419	-0.2060474	NA	0.0078382	[illegible]	NA	NA	NA
preserved ~ pos	1518.[illegible]	[illegible]	NA	NA	-0.2297194	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	1519.[illegible]	-0.0531501	NA	NA	-0.2108225	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	1536.[illegible]	-0.1402533	0.1267[illegible]	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	1538.[illegible]	-0.1408018	0.1612835	-0.0042574	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	1545.[illegible]	-0.1792198	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	1559.[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     5.59913     -0.04965      0.05919     -0.77181      0.12594
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4430 Residual
## Null Deviance:       1433
## Residual Deviance: 1366  AIC: 1497
```

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

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

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

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

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

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

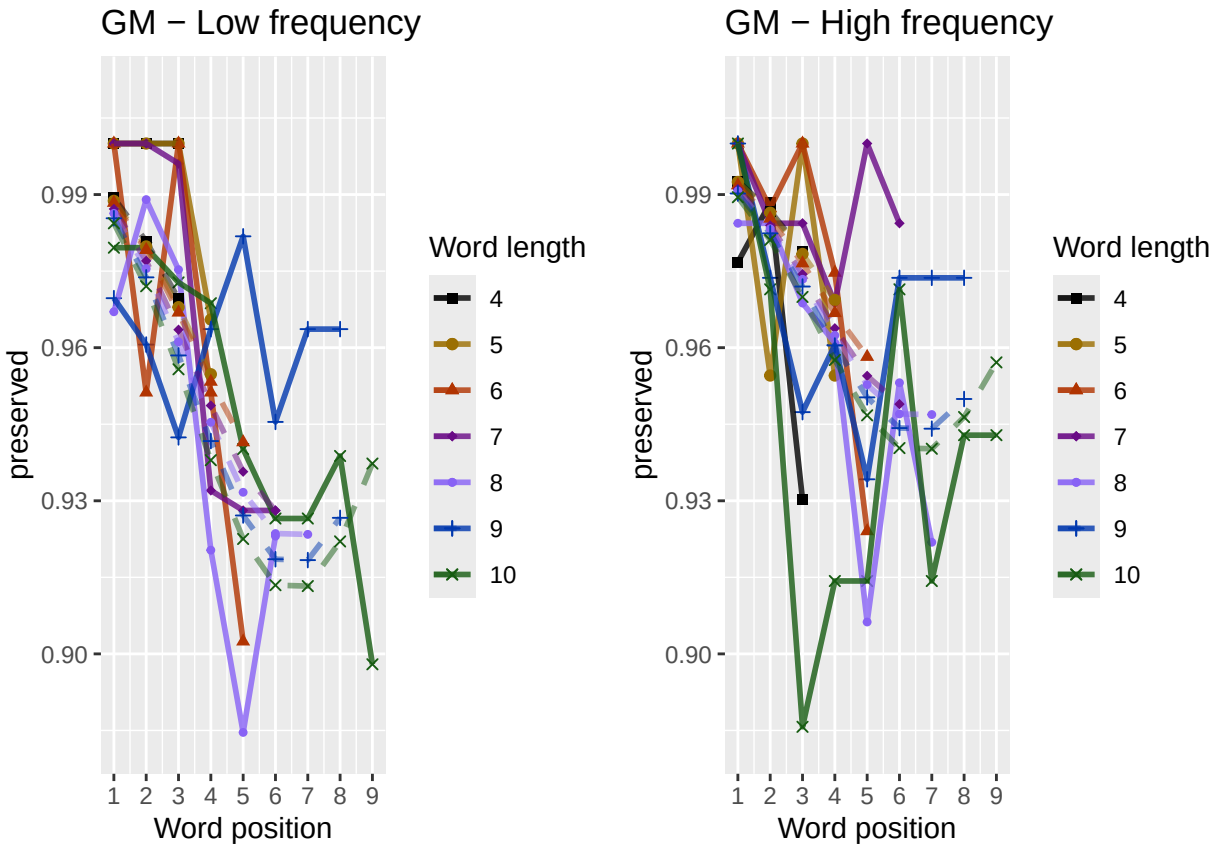
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

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

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

```
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
##       3.617       -1.601
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1188  AIC: 1294
## log likelihood:  -593.9163
## Nagelkerke R2:  0.1946049
## % pres/err predicted correctly:  -284.0179
## % of predictable range [ (model-null)/(1-null) ]:  0.1534504
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      5.2364       0.0553      -0.7653
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1378  AIC: 1507
## log likelihood:  -688.8618
## Nagelkerke R2:  0.04465363
## % pres/err predicted correctly:  -331.7343
## % of predictable range [ (model-null)/(1-null) ]:  0.01172487
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      4.1820      -0.2297
##
```

```
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1389  AIC: 1519
## log likelihood:  -694.4637
## Nagelkerke R2:  0.03560404
## % pres/err predicted correctly:  -332.4857
## % of predictable range [ (model-null)/(1-null) ]:  0.009493139
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.6004      -0.1792
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1418  AIC: 1546
## log likelihood:  -709.2487
## Nagelkerke R2:  0.01160977
## % pres/err predicted correctly:  -334.4819
## % of predictable range [ (model-null)/(1-null) ]:  0.003563977
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     3.37131     -0.06438
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1430  AIC: 1559
## log likelihood:  -714.7983
## Nagelkerke R2:  0.002562108
## % pres/err predicted correctly:  -335.5269
## % of predictable range [ (model-null)/(1-null) ]:  0.0004601546
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.186
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4434 Residual
## Null Deviance:       1433
## Residual Deviance: 1433  AIC: 1559
```

```
## log likelihood:  -716.3673
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -335.6818
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1293.991	0.0000	1	1	0.1946049	3.617141	NA	-1.601379	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1506.886	212.8946	0	0	0.0446536	5.236382	NA	NA	0.0552979	-0.7653311	NA
preserved ~ pos	1518.642	224.6504	0	0	0.0356040	4.181985	NA	NA	NA	-0.2297194	NA
preserved ~ stimlen	1545.799	251.8080	0	0	0.0116098	4.600376	NA	NA	NA	NA	-0.1792198
preserved ~ CumPres	1559.032	265.0410	0	0	0.0025623	3.371310	-0.0643793	NA	NA	NA	NA
preserved ~ 1	1559.226	265.2345	0	0	0.0000000	3.186306	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	0.9685185	585
O	0.9492835	2047
P	1.0000000	36
S	0.9570312	256
V	0.9716964	1511

```
#  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
##       3.591       -1.585
##
## Degrees of Freedom: 4142 Total (i.e. Null);  4141 Residual
## Null Deviance:       1339
## Residual Deviance: 1121  AIC: 1227
## log likelihood:  -560.467
## Nagelkerke R2:  0.1856507
## % pres/err predicted correctly:  -268.5009
## % of predictable range [ (model-null)/(1-null) ]:  0.1458062
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     5.11610      0.05328     -0.72847
##
## Degrees of Freedom: 4142 Total (i.e. Null);  4140 Residual
## Null Deviance:       1339
## Residual Deviance: 1293  AIC: 1422
## log likelihood:  -646.5915
## Nagelkerke R2:  0.03981142
## % pres/err predicted correctly:  -311.2121
## % of predictable range [ (model-null)/(1-null) ]:  0.01043142
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      4.1142      -0.2152
##
## Degrees of Freedom: 4142 Total (i.e. Null);  4141 Residual
## Null Deviance:       1339
## Residual Deviance: 1303  AIC: 1433
## log likelihood:  -651.4983
## Nagelkerke R2:  0.03131844
## % pres/err predicted correctly:  -311.894
## % of predictable range [ (model-null)/(1-null) ]:  0.008270251
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen
##      4.6515      -0.1859
##
## Degrees of Freedom: 4142 Total (i.e. Null);  4141 Residual
## Null Deviance:       1339
## Residual Deviance: 1325  AIC: 1452
## log likelihood:  -662.3303
## Nagelkerke R2:  0.01249862
## % pres/err predicted correctly:  -313.2742
## % of predictable range [ (model-null)/(1-null) ]:  0.003895678
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     3.36906     -0.06936
##
## Degrees of Freedom: 4142 Total (i.e. Null);  4141 Residual
## Null Deviance:       1339
## Residual Deviance: 1336  AIC: 1465
## log likelihood:  -667.9262
## Nagelkerke R2:  0.002737381
## % pres/err predicted correctly:  -314.337
## % of predictable range [ (model-null)/(1-null) ]:  0.0005268831
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.183
##
## Degrees of Freedom: 4142 Total (i.e. Null);  4142 Residual
## Null Deviance:       1339
## Residual Deviance: 1339  AIC: 1465
## log likelihood:  -669.4928
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -314.5033
## % 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	1226.797	0.0000	1	1	0.1856507	3.590596	NA	-1.58465	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1422.182	195.3854	0	0	0.0398114	5.116095	NA	NA	0.0532769	-0.7284746	NA
preserved ~ pos	1432.514	205.7168	0	0	0.0313184	4.114241	NA	NA	NA	-0.2152439	NA
preserved ~ stimlen	1451.884	225.0867	0	0	0.0124986	6.651458	NA	NA	NA	NA	-0.1858763
preserved ~ CumPres	1465.103	238.3057	0	0	0.0027373	3.369057	-0.0693555	NA	NA	NA	NA
preserved ~ 1	1465.388	238.5907	0	0	0.0000000	3.182589	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
##       3.482       -1.683
##
## Degrees of Freedom: 3557 Total (i.e. Null);  3556 Residual
## Null Deviance:       1183
## Residual Deviance: 1043  AIC: 1127
## log likelihood:  -521.2894
## Nagelkerke R2:  0.1371205
## % pres/err predicted correctly:  -251.1075
## % of predictable range [ (model-null)/(1-null) ]:  0.09956214
## ***************************
```

```
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     5.18585      0.05805     -0.77922
##
## Degrees of Freedom: 3557 Total (i.e. Null);  3555 Residual
## Null Deviance:       1183
## Residual Deviance: 1137  AIC: 1238
## log likelihood:  -568.474
## Nagelkerke R2:  0.04581019
## % pres/err predicted correctly:  -275.5104
## % of predictable range [ (model-null)/(1-null) ]:  0.01240362
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      4.1046      -0.2203
##
## Degrees of Freedom: 3557 Total (i.e. Null);  3556 Residual
## Null Deviance:       1183
## Residual Deviance: 1148  AIC: 1250
## log likelihood:  -574.0161
## Nagelkerke R2:  0.03492531
## % pres/err predicted correctly:  -276.3177
## % of predictable range [ (model-null)/(1-null) ]:  0.009520352
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.4409      -0.1646
##
## Degrees of Freedom: 3557 Total (i.e. Null);  3556 Residual
## Null Deviance:       1183
## Residual Deviance: 1173  AIC: 1274
## log likelihood:  -586.5779
## Nagelkerke R2:  0.01012796
## % pres/err predicted correctly:  -278.0433
## % of predictable range [ (model-null)/(1-null) ]:  0.003356978
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      3.4446      -0.1276
##
## Degrees of Freedom: 3557 Total (i.e. Null);  3556 Residual
## Null Deviance:       1183
## Residual Deviance: 1176  AIC: 1279
## log likelihood:  -588.0227
## Nagelkerke R2:  0.007264755
## % pres/err predicted correctly:  -278.4618
## % of predictable range [ (model-null)/(1-null) ]:  0.001862355
## ****************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.147
##
## Degrees of Freedom: 3557 Total (i.e. Null);  3557 Residual
## Null Deviance:       1183
## Residual Deviance: 1183  AIC: 1283
## log likelihood:  -591.6831
## Nagelkerke R2:  7.847816e-16
## % pres/err predicted correctly:  -278.9832
## % of predictable range [ (model-null)/(1-null) ]:  0
## ****************************
```

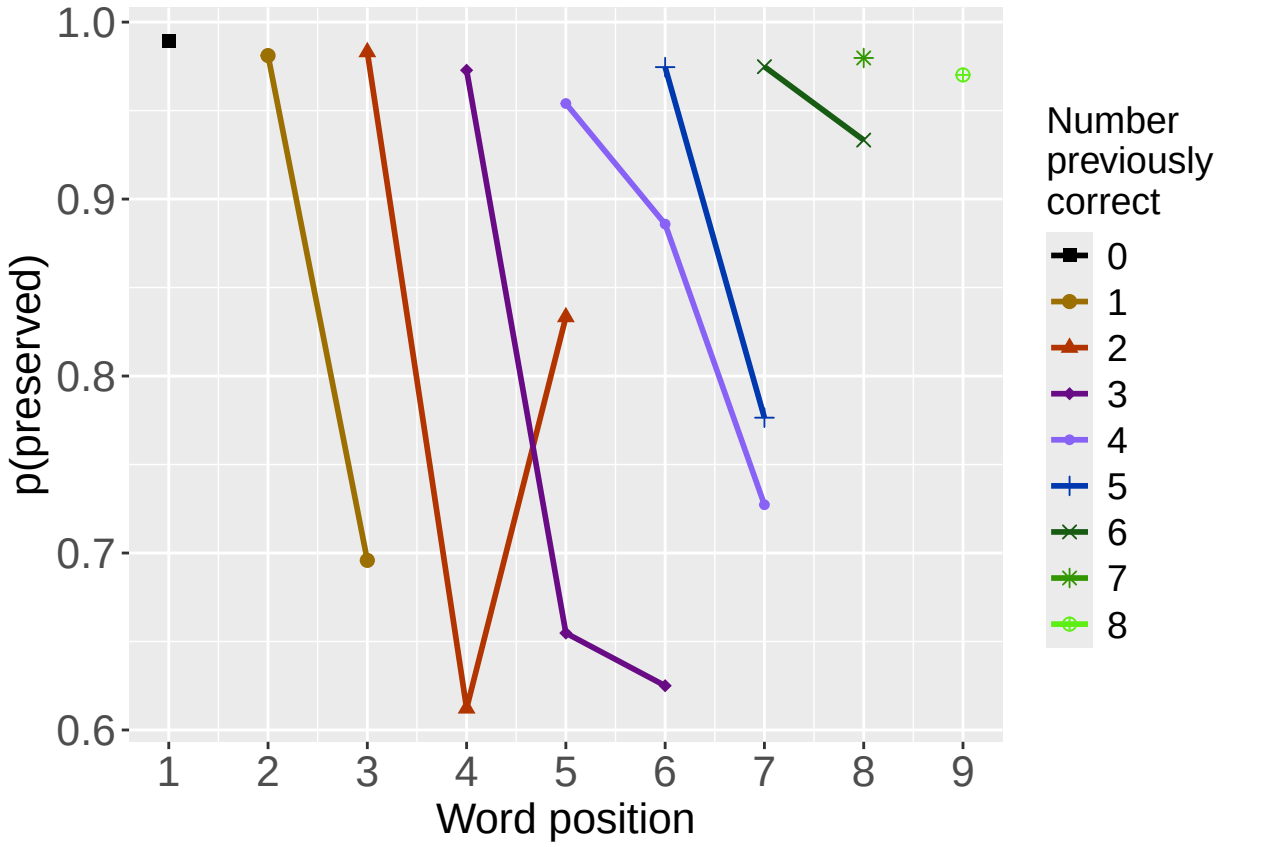
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1126.626	0.0000	1	1	0.137120	3.481957	NA	-1.683304	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1238.053	111.4269	0	0	0.045810	2.185847	NA	NA	0.0580526	-0.7792226	NA
preserved ~ pos	1249.915	123.2889	0	0	0.034925	3.104636	NA	NA	NA	-0.2203463	NA
preserved ~ stimlen	1273.645	147.0192	0	0	0.010128	0.440887	NA	NA	NA	NA	-0.1645763
preserved ~ CumPres	1278.570	151.9447	0	0	0.007264	3.444596	-0.1276402	NA	NA	NA	NA
preserved ~ 1	1283.331	156.7053	0	0	0.000000	3.147289	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.
```

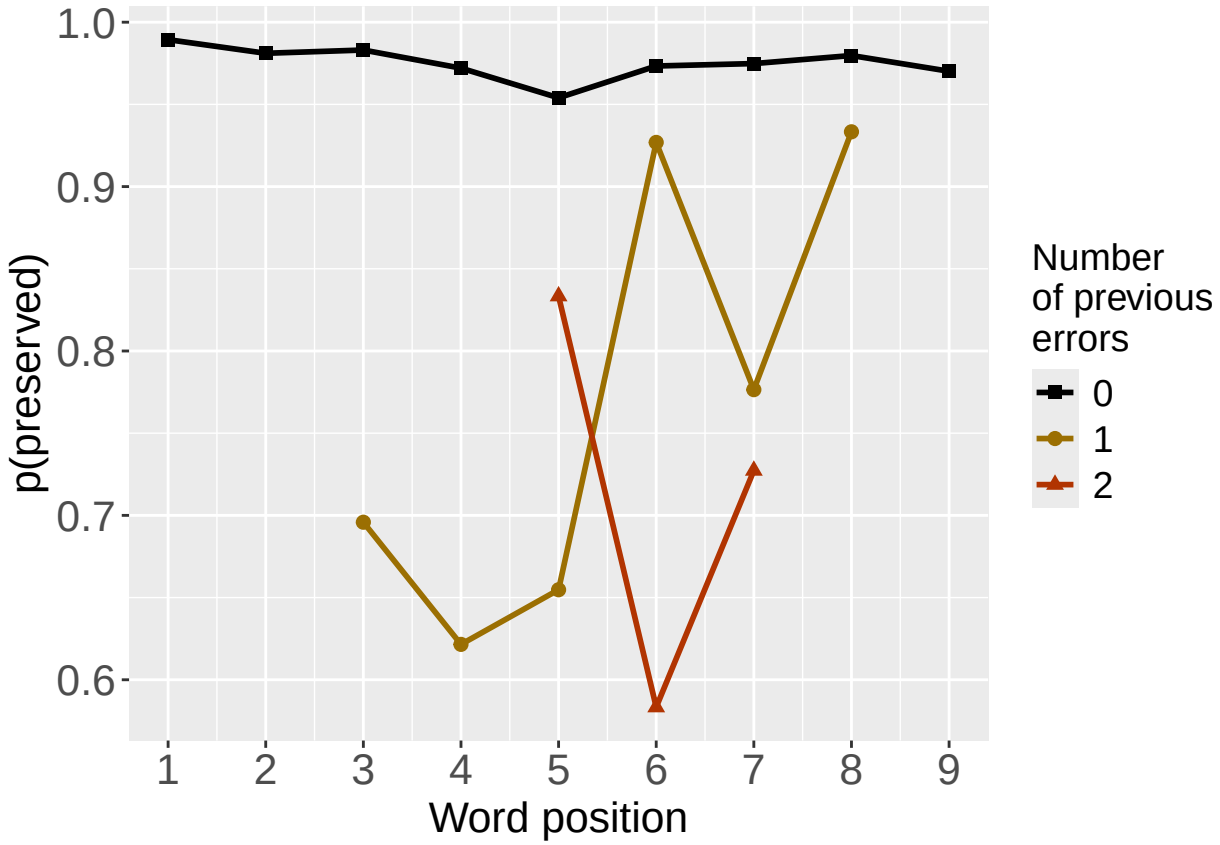
```
print(PrevCorPlot)
```




```
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
```

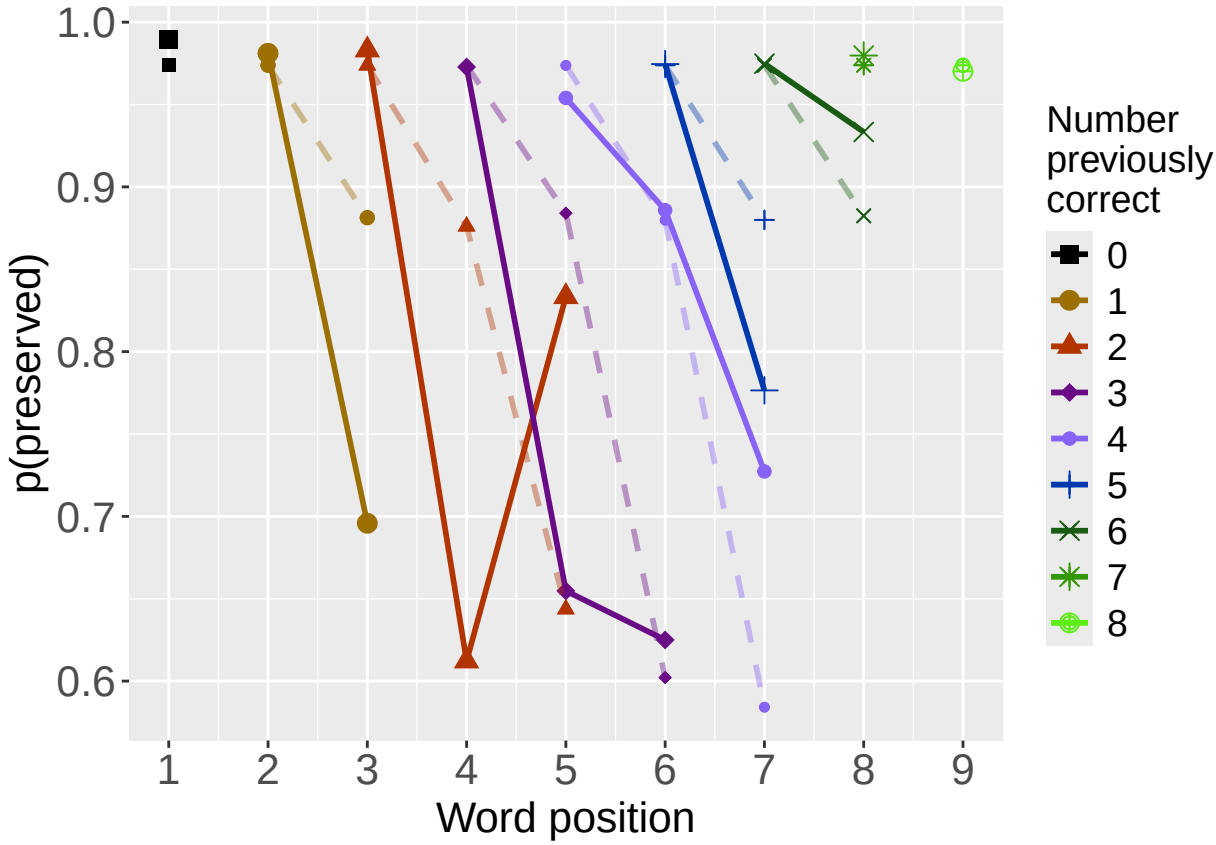
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

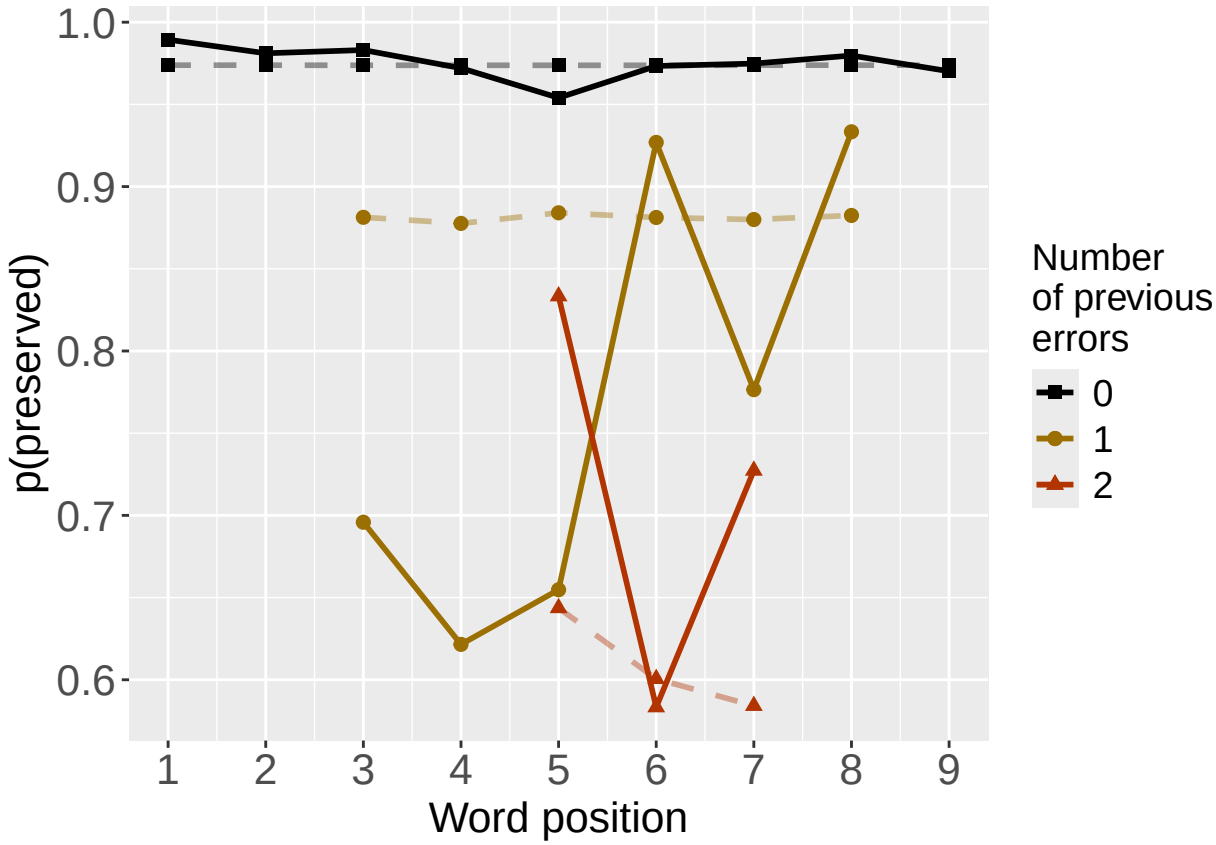
```
print(PrevCorPlot)
```




```
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
##     5.26513     -1.58269      0.08366     -0.82559
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1165  AIC: 1272
## log likelihood:  -582.67
## Nagelkerke R2:  0.211945
## % pres/err predicted correctly:  -280.495
## % of predictable range [ (model-null)/(1-null) ]:  0.1639138
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.617       -1.601
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1188  AIC: 1294
## log likelihood:  -593.9163
## Nagelkerke R2:  0.1946049
## % pres/err predicted correctly:  -284.0179
## % of predictable range [ (model-null)/(1-null) ]:  0.1534504
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      5.2364       0.0553      -0.7653
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1378  AIC: 1507
## log likelihood:  -688.8618
## Nagelkerke R2:  0.04465363
## % pres/err predicted correctly:  -331.7343
## % of predictable range [ (model-null)/(1-null) ]:  0.01172487
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1271.779	0.00000	1.0e+00	0.999985	0.2119450	5.265130	-1.582687	0.0836569	-0.8255862

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1293.991	22.21234	1.5e-05	0.000015	0.1946049	3.617141	-1.601379	NA	NA
preserved ~ I(posˆ2) + pos	1506.886	235.10699	0.0e+00	0.000000	0.0446536	5.236382	NA	0.0552979	-0.7653311

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.617       -1.601
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1188  AIC: 1294
## log likelihood:  -593.9163
## Nagelkerke R2:  0.1946049
## % pres/err predicted correctly:  -284.0179
## % of predictable range [ (model-null)/(1-null) ]:  0.1534504
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     3.85488     -1.58286     -0.03101
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1187  AIC: 1296
## log likelihood:  -593.7447
## Nagelkerke R2:  0.1948702
## % pres/err predicted correctly:  -284.1581
## % of predictable range [ (model-null)/(1-null) ]:  0.153034
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.6004      -0.1792
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1418  AIC: 1546
## log likelihood:  -709.2487
## Nagelkerke R2:  0.01160977
## % pres/err predicted correctly:  -334.4819
## % of predictable range [ (model-null)/(1-null) ]:  0.003563977
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	1293.991	0.000000	1.0000000	0.6967007	0.1946049	3.617141	-1.601379	NA
preserved ~ CumErr + stimlen	1295.655	1.663272	0.4353365	0.3032993	0.1948702	3.854882	-1.582865	-0.0310052
preserved ~ stimlen	1545.799	251.807980	0.0000000	0.0000000	0.0116098	4.600376	NA	-0.1792198

```
########
# 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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.617       -1.601
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
```

```
## Null Deviance:       1433
## Residual Deviance: 1188  AIC: 1294
## log likelihood:  -593.9163
## Nagelkerke R2:  0.1946049
## % pres/err predicted correctly:  -284.0179
## % of predictable range [ (model-null)/(1-null) ]:  0.1534504
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     3.78171     -1.58664     -0.05718
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1186  AIC: 1295
## log likelihood:  -592.963
## Nagelkerke R2:  0.1960781
## % pres/err predicted correctly:  -284.5793
## % of predictable range [ (model-null)/(1-null) ]:  0.151783
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     3.37131     -0.06438
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1430  AIC: 1559
## log likelihood:  -714.7983
## Nagelkerke R2:  0.002562108
## % pres/err predicted correctly:  -335.5269
## % of predictable range [ (model-null)/(1-null) ]:  0.0004601546
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr	1293.991	0.000000	1.0000000	0.6501218	0.1946049	3.617141	-1.601379	NA
preserved ~ CumErr + CumPres	1295.230	1.239149	0.5381734	0.3498782	0.1960781	3.781713	-1.586640	-0.0571806
preserved ~ CumPres	1559.032	265.041004	0.0000000	0.0000000	0.0025621	3.371310	NA	-0.0643793

```
########
# 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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.617       -1.601
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1188  AIC: 1294
## log likelihood:  -593.9163
## Nagelkerke R2:  0.1946049
## % pres/err predicted correctly:  -284.0179
## % of predictable range [ (model-null)/(1-null) ]:  0.1534504
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     3.83889     -1.52946     -0.05718
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1186  AIC: 1295
## log likelihood:  -592.963
## Nagelkerke R2:  0.1960781
## % pres/err predicted correctly:  -284.5793
## % of predictable range [ (model-null)/(1-null) ]:  0.151783
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      4.1820      -0.2297
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1389  AIC: 1519
## log likelihood:  -694.4637
## Nagelkerke R2:  0.03560404
## % pres/err predicted correctly:  -332.4857
## % of predictable range [ (model-null)/(1-null) ]:  0.009493139
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr	1293.991	0.000000	1.0000000	0.6501218	0.1946049	3.617141	-1.601379	NA
preserved ~ CumErr + pos	1295.230	1.239149	0.5381734	0.3498782	0.1960781	3.838894	-1.529460	-0.0571806
preserved ~ pos	1518.642	224.650352	0.0000000	0.0000000	0.0356040	4.181985	NA	-0.2297194

```
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	1271.779	0.000000	1.000000	0.9999850	0.2119455	3.626513	-1.582687	0.0836569	-0.8255862	NA	NA
preserved ~ CumErr	1293.991	22.212340	0.000015	0.0000015	0.1946049	3.617141	-1.601379	NA	NA	NA	NA
preserved ~ CumErr	1293.990	0.000000	1.000000	0.6967007	0.1946049	3.617141	-1.601379	NA	NA	NA	NA
preserved ~ CumErr	1293.990	0.000000	1.000000	0.6501218	0.1946049	3.617141	-1.601379	NA	NA	NA	NA
preserved ~ CumErr	1293.990	0.000000	1.000000	0.6501218	0.1946049	3.617141	-1.601379	NA	NA	NA	NA
preserved ~ CumErr + CumPres	1295.230	1.239149	0.538173	0.3498782	0.1960781	3.781713	-1.586640	NA	NA	NA	-0.0571806
preserved ~ CumErr + pos	1295.230	1.239149	0.538173	0.3498782	0.1960781	3.838894	-1.529460	NA	-0.0571806	NA	NA
preserved ~ CumErr + stimlen	1295.655	1.663272	0.435336	0.2830299	0.1948703	3.854882	-1.582865	NA	NA	-0.0310052	NA
preserved ~ I(pos^2) + pos	1506.882	235.106985	0.000000	0.0000000	0.0044653	6.236382	NA	0.0552979	-0.7653311	NA	NA
preserved ~ pos	1518.642	224.650352	0.000000	0.0000000	0.0356040	4.181985	NA	NA	-0.2297194	NA	NA
preserved ~ stimlen	1545.792	51.807980	0.000000	0.0000000	0.0116098	8.600376	NA	NA	NA	-0.1792198	NA
preserved ~ CumPres	1559.032	65.041004	0.000000	0.0000000	0.0025623	4.371310	NA	NA	NA	NA	-0.0643793

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     5.27087     -1.56420      0.08535     -0.82876      0.12094
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4430 Residual
## Null Deviance:       1433
## Residual Deviance: 1158  AIC: 1264
## log likelihood:  -578.9115
## Nagelkerke R2:  0.2177207
## % pres/err predicted correctly:  -280.0787
## % of predictable range [ (model-null)/(1-null) ]:  0.1651505
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##    5.313272    -1.563255     0.085576    -0.829209    -0.005884     0.119954
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4429 Residual
## Null Deviance:       1433
## Residual Deviance: 1158  AIC: 1266
## log likelihood:  -578.9068
## Nagelkerke R2:  0.2177278
## % pres/err predicted correctly:  -280.084
## % of predictable range [ (model-null)/(1-null) ]:  0.1651347
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     5.26513     -1.58269      0.08366     -0.82559
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1165  AIC: 1272
## log likelihood:  -582.67
## Nagelkerke R2:  0.211945
## % pres/err predicted correctly:  -280.495
## % of predictable range [ (model-null)/(1-null) ]:  0.1639138
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     5.56455     -1.57495      0.08541     -0.82864     -0.04180
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4430 Residual
## Null Deviance:       1433
## Residual Deviance: 1165  AIC: 1273
## log likelihood:  -582.4209
## Nagelkerke R2:  0.2123283
## % pres/err predicted correctly:  -280.5095
## % of predictable range [ (model-null)/(1-null) ]:  0.163871
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       3.186
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4434 Residual
## Null Deviance:       1433
## Residual Deviance: 1433  AIC: 1559
## log likelihood:  -716.3673
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -335.6818
## % 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 + log_freq	1264.357	0.000000	1.0000000	0.7071649	0.2177257	5.270871	-1.564195	0.0853460	-0.8287580	0.1209401	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	1266.314	1.957242	0.3758290	0.2657731	0.2177257	5.813272	-1.563255	0.0855759	-0.8292090	0.1199542	-0.0058837
preserved ~ CumErr + I(pos^2) + pos	1271.779	7.422155	0.0244512	0.0172901	0.2119450	5.265130	-1.582687	0.0836569	-0.8255862	NA	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen	1272.921	8.563700	0.0138171	0.0097710	0.2123283	5.564551	-1.574952	0.0854053	-0.8286447	NA	-0.0418001
preserved ~ 1	1559.226	294.869027	0.0000000	0.0000000	0.0000000	3.186306	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!
```

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

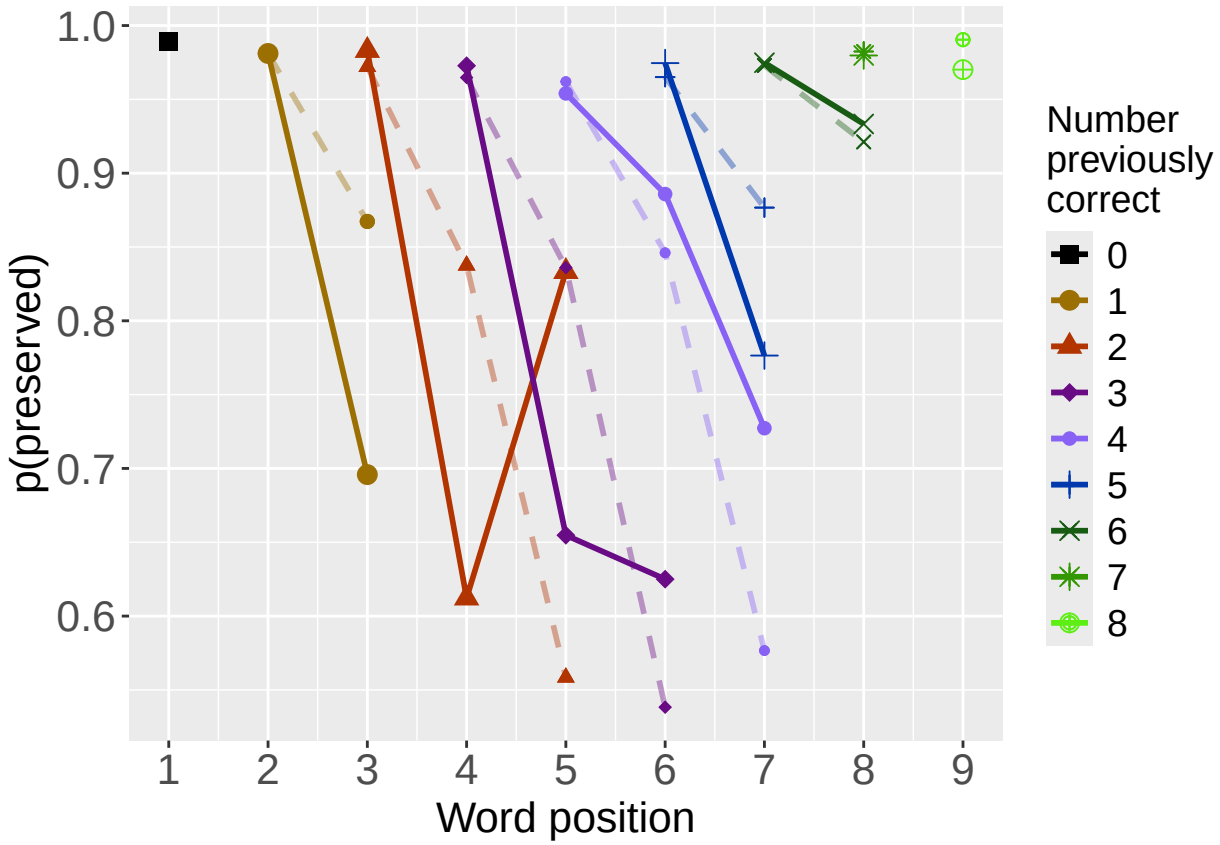
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + log_freq
##          Df Deviance    AIC
## CumErr    1   1367.0 1471.6
## pos       1   1180.5 1285.0
## I(pos^2)  1   1179.2 1283.8
## log_freq  1   1165.3 1269.9
## <none>        1157.8 1264.4
```

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

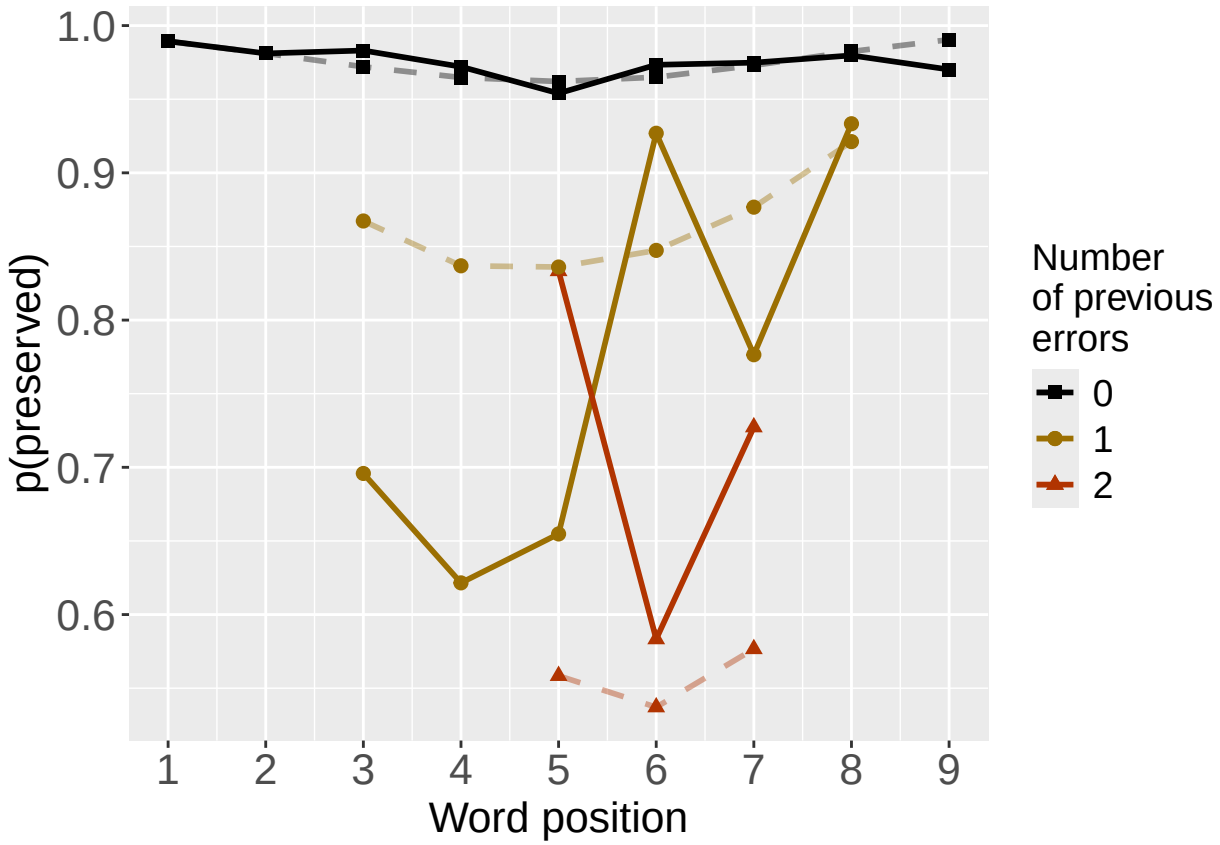
```
print(PrevCorPlot)
```




```
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!
```

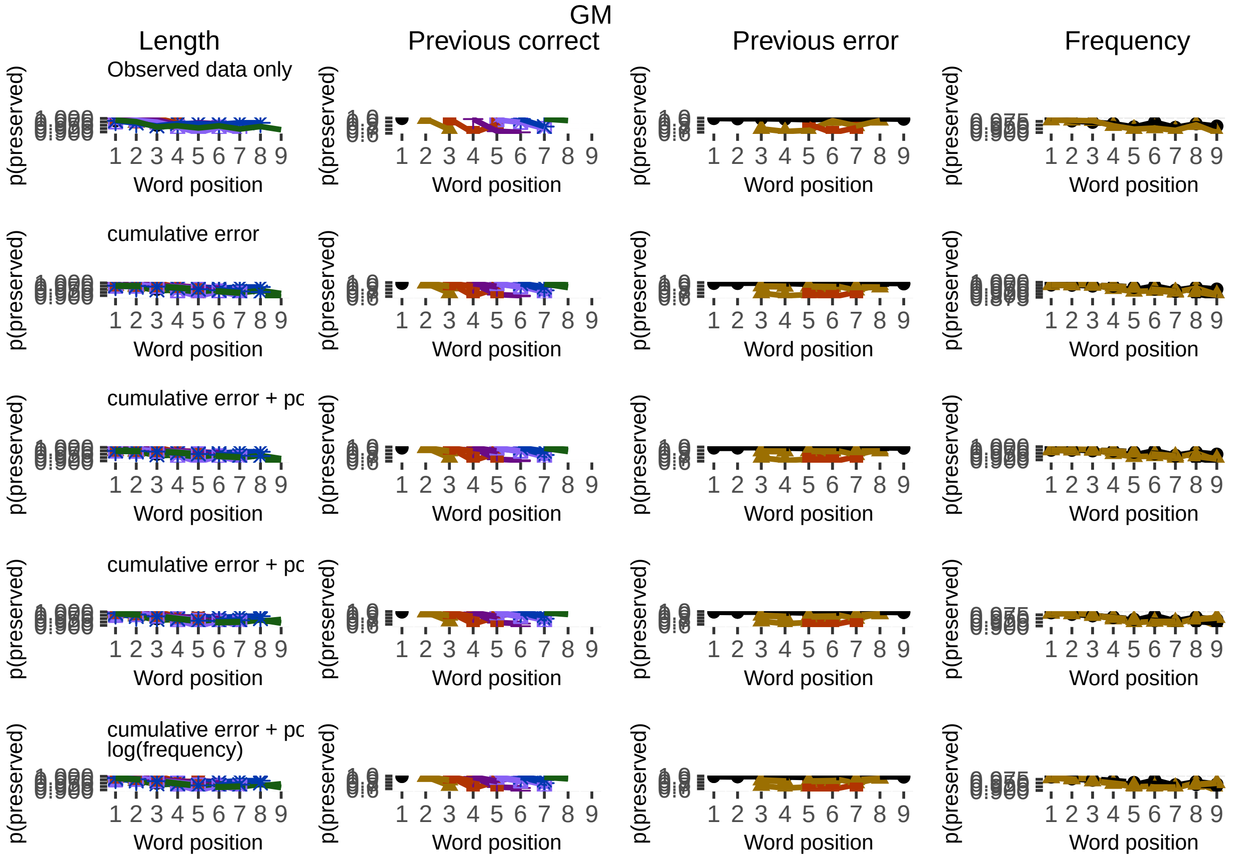
```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     3.83889     -1.52946     -0.05718
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4432 Residual
## Null Deviance:       1433
## Residual Deviance: 1186  AIC: 1295
## log likelihood:  -592.963
```

```
## Nagelkerke R2:  0.1960781
## % pres/err predicted correctly:  -284.5793
## % of predictable range [ (model-null)/(1-null) ]:  0.151783
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.617       -1.601
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4433 Residual
## Null Deviance:       1433
## Residual Deviance: 1188  AIC: 1294
## log likelihood:  -593.9163
## Nagelkerke R2:  0.1946049
## % pres/err predicted correctly:  -284.0179
## % of predictable range [ (model-null)/(1-null) ]:  0.1534504
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)
##     5.26513     -1.58269     -0.82559      0.08366
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4431 Residual
## Null Deviance:       1433
## Residual Deviance: 1165  AIC: 1272
## log likelihood:  -582.67
## Nagelkerke R2:  0.211945
## % pres/err predicted correctly:  -280.495
## % of predictable range [ (model-null)/(1-null) ]:  0.1639138
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)     log_freq
##     5.27087     -1.56420     -0.82876      0.08535      0.12094
##
## Degrees of Freedom: 4434 Total (i.e. Null);  4430 Residual
## Null Deviance:       1433
## Residual Deviance: 1158  AIC: 1264
## log likelihood:  -578.9115
## Nagelkerke R2:  0.2177207
## % pres/err predicted correctly:  -280.0787
## % of predictable range [ (model-null)/(1-null) ]:  0.1651505
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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




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

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

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

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

	CumErr	I(pos^2)	pos	log_freq
McFadden	0.1558864	0.0133283	0.0173353	0.0079426
SquaredCorrelation	0.0528237	0.0045404	0.0059324	0.0027145
Nagelkerke	0.0528237	0.0045404	0.0059324	0.0027145
Estrella	0.0587631	0.0049889	0.0064252	0.0029526

```
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 1157.823
## CumErr + pos + I(pos^2)                       CumErr + pos + I(pos^2) 1165.340
## CumErr + pos                                            CumErr + pos 1185.926
## CumErr                                                        CumErr 1187.833
## null                                                            null 1432.735
##                                    deviance_explained percent_explained
## CumErr + pos + I(pos^2) + log_freq           274.9117          19.18790
## CumErr + pos + I(pos^2)                      267.3945          18.66323
## CumErr + pos                                 246.8085          17.22640
## CumErr                                       244.9020          17.09333
## null                                           0.0000           0.00000
##                                    percent_of_explained_deviance increment_in_explained
## CumErr + pos + I(pos^2) + log_freq                     100.00000              2.7344011
## CumErr + pos + I(pos^2)                                 97.26560              7.4882121
## CumErr + pos                                            89.77739              0.6935078
## CumErr                                                  89.08388             89.0838791
## null                                                          NA              0.0000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + I(pos^2) + log_freq	1157.823	274.9117
CumErr + pos + I(pos^2)	1165.340	267.3945
CumErr + pos	1185.926	246.8085
CumErr	1187.833	244.9020
null	1432.735	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + I(pos^2) + log_freq	19.18790	100.00000	2.7344011
CumErr + pos + I(pos^2)	18.66323	97.26560	7.4882121
CumErr + pos	17.22640	89.77739	0.6935078
CumErr	17.09333	89.08388	89.0838791
null	0.00000	NA	0.0000000

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

```
##          Nagelkerke
## CumErr   0.80022603
## I(pos^2) 0.06878252
## pos      0.08986913
## log_freq 0.04112232
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr+pos	0.8461391	1185.926
preserved ~ CumErr	0.8546343	1187.833
preserved ~ CumErr+pos+I(pos^2)	0.8835124	1165.340
preserved ~ CumErr+pos+I(pos^2)+log_freq	0.8868558	1157.823

```
## 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       0.8461391       1185.926     0.000000000
## 2                       preserved ~ CumErr       0.8546343       1187.833     0.008495247
## 3          preserved ~ CumErr+pos+I(pos^2)       0.8835124       1165.340     0.037373336
## 4 preserved ~ CumErr+pos+I(pos^2)+log_freq       0.8868558       1157.823     0.040716683
##    diff_CumErr diff_CumErr+pos+I(pos^2) diff_CumErr+pos+I(pos^2)+log_freq
## 1 -0.008495247             -0.037373336                      -0.040716683
## 2  0.000000000             -0.028878089                      -0.032221436
## 3  0.028878089              0.000000000                      -0.003343347
## 4  0.032221436              0.003343347                       0.000000000
```

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

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

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

model	diff_CumErr+pos	diff_CumErr	diff_CumErr+pos+I(pos^2)
preserved ~ CumErr+pos	0.0000000	-0.0084952	-0.0373733
preserved ~ CumErr	0.0084952	0.0000000	-0.0288781
preserved ~ CumErr+pos+I(pos^2)	0.0373733	0.0288781	0.0000000
preserved ~ CumErr+pos+I(pos^2)+log_freq	0.0407167	0.0322214	0.0033433