DG - repetition - Serial position analysis (v5)

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

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

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

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

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

```r
# 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)
```

```r
# 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	549	35	131	NA	NA	715
2	67	NA	437	99	112	715
3	314	NA	175	211	15	715
4	307	NA	241	69	39	656
5	235	NA	216	72	39	562
6	208	1	139	74	22	444
7	180	NA	105	29	18	332
8	93	NA	55	26	4	178
9	76	NA	2	NA	7	85

```r
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7678322	0.0489510	0.1832168	NA	NA	715
2	0.0937063	NA	0.6111888	0.1384615	0.1566434	715
3	0.4391608	NA	0.2447552	0.2951049	0.0209790	715
4	0.4679878	NA	0.3673780	0.1051829	0.0594512	656
5	0.4181495	NA	0.3843416	0.1281139	0.0693950	562
6	0.4684685	0.0022523	0.3130631	0.1666667	0.0495495	444

pos_factor	O	P	V	1	S	total
7	0.5421687	NA	0.3162651	0.0873494	0.0542169	332
8	0.5224719	NA	0.3089888	0.1460674	0.0224719	178
9	0.8941176	NA	0.0235294	NA	0.0823529	85

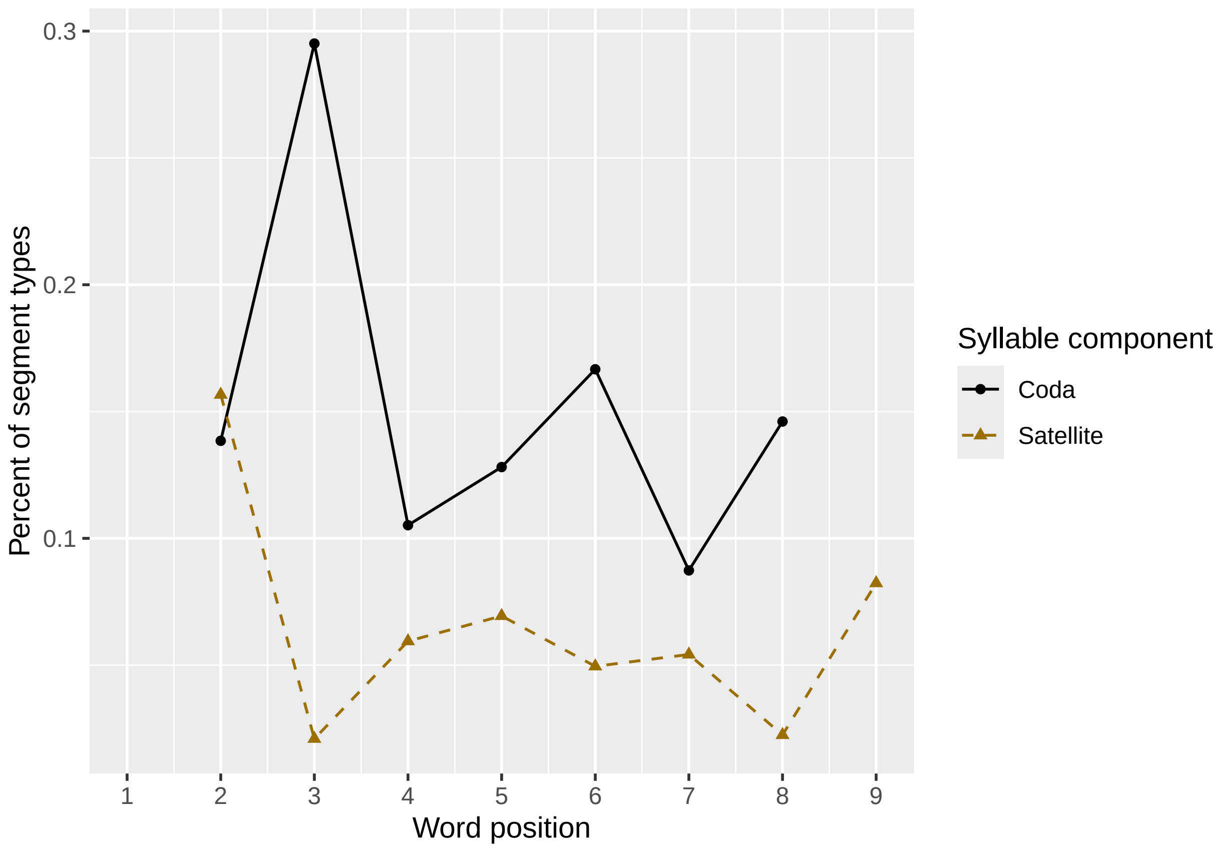
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 0.915 0.966 0.898 NA     NA     NA     NA     NA     NA
## 2       5 0.883 0.915 0.941  0.910 NA     NA     NA     NA     NA
## 3       6 0.958 0.922 0.873  0.886  0.895 NA     NA     NA     NA
## 4       7 0.911 0.911 0.875  0.847  0.905  0.865 NA     NA     NA
## 5       8 0.938 0.906 0.892  0.865  0.844  0.857  0.853 NA     NA
## 6       9 0.957 0.878 0.871  0.857  0.796  0.760  0.855  0.753 NA
## 7      10 0.953 0.910 0.872  0.822  0.763  0.722  0.789  0.880  0.798
```

```
# 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.
```

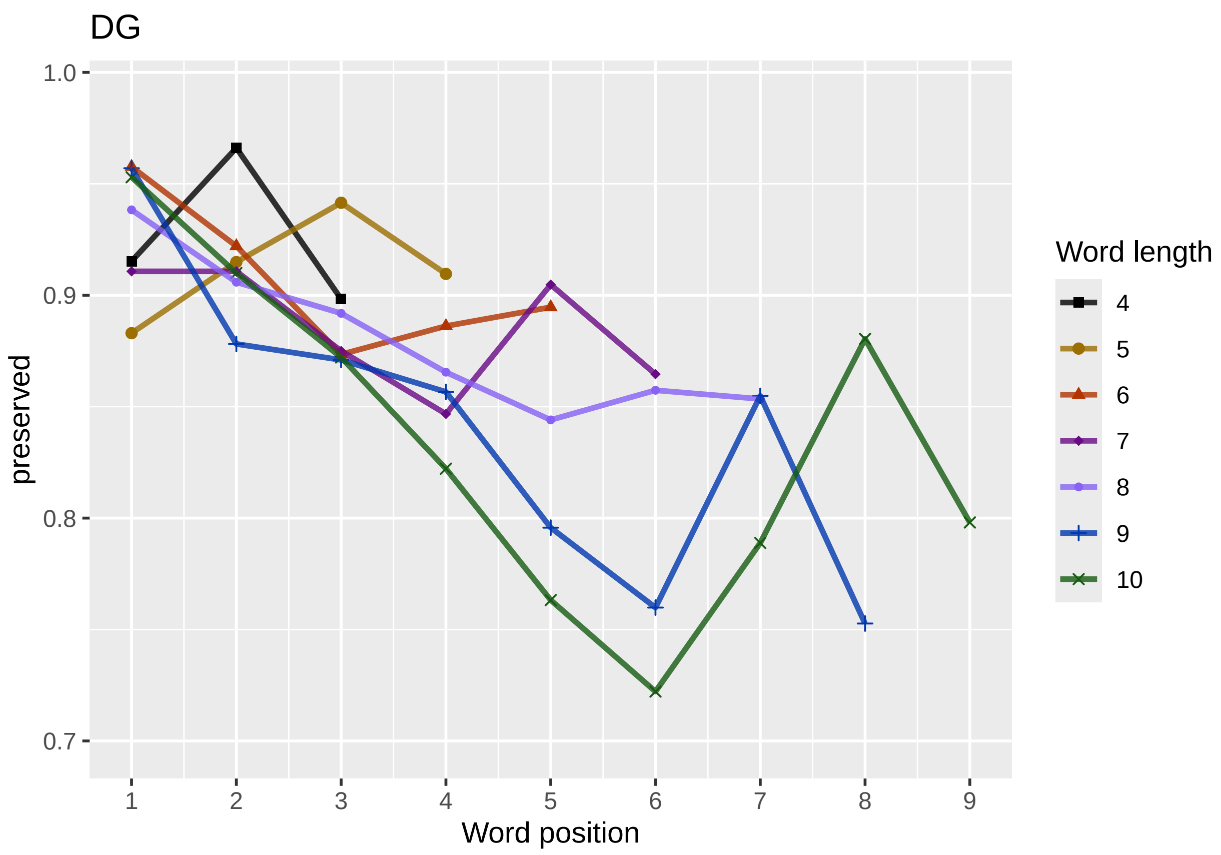
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    59    59    59    NA    NA    NA    NA    NA    NA
## 2       5    94    94    94    94    NA    NA    NA    NA    NA
## 3       6   118   118   118   118   118    NA    NA    NA    NA
## 4       7   112   112   112   112   112   112    NA    NA    NA
## 5       8   154   154   154   154   154   154   154    NA    NA
## 6       9    93    93    93    93    93    93    93    93    NA
## 7      10    85    85    85    85    85    85    85    85    85
```

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

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

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



Length and position

```
# length and position

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

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

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

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

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          1.40765           0.21412          -0.04578           0.62976           0.01065
##      stimlen:pos
##         -0.13915
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4396 Residual
## Null Deviance:       3094
## Residual Deviance: 3006  AIC: 3452
## log likelihood:  -1502.856
## Nagelkerke R2:  0.03936943
## % pres/err predicted correctly:  -920.4609
## % of predictable range [ (model-null)/(1-null) ]:  0.02500353
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.74421     -0.10451      0.02794     -0.38465
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 3017  AIC: 3458
## log likelihood:  -1508.575
## Nagelkerke R2:  0.0343176
## % pres/err predicted correctly:  -923.3998
## % of predictable range [ (model-null)/(1-null) ]:  0.0218939
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.97511      0.02255     -0.36881
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3028  AIC: 3467
## log likelihood:  -1513.961
## Nagelkerke R2:  0.02954877
## % pres/err predicted correctly:  -926.9392
## % of predictable range [ (model-null)/(1-null) ]:  0.01814887
## ***************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.17688     -0.08804     -0.13268
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3026  AIC: 3471
## log likelihood:  -1512.934
## Nagelkerke R2:  0.0304596
## % pres/err predicted correctly:  -925.837
## % of predictable range [ (model-null)/(1-null) ]:  0.01931511
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.92819     -0.05842     -0.05900     -0.00850
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 3026  AIC: 3473
## log likelihood:  -1512.763
## Nagelkerke R2:  0.03061047
## % pres/err predicted correctly:  -925.7679
## % of predictable range [ (model-null)/(1-null) ]:  0.01938816
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.6044      -0.1613
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3034  AIC: 3476
## log likelihood:  -1516.879
## Nagelkerke R2:  0.02696006
## % pres/err predicted correctly:  -928.3171
## % of predictable range [ (model-null)/(1-null) ]:  0.0166909
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      3.1979      -0.1603
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3061  AIC: 3510
## log likelihood:  -1530.394
## Nagelkerke R2:  0.01492581
## % pres/err predicted correctly:  -935.1915
## % of predictable range [ (model-null)/(1-null) ]:  0.00941706
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.94
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4401 Residual
## Null Deviance:       3094
## Residual Deviance: 3094  AIC: 3543
## log likelihood:  -1547.042
## Nagelkerke R2:  4.398286e-16
## % pres/err predicted correctly:  -944.0915
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	3452.414	0.000000	1.0000000	0.9402101	0.0393691	4.4076540	0.2141152	0.6297562	-0.1391490	-0.0457833	0.0106536
preserved ~ stimlen + I(pos^2) + pos	3457.949	5.535081	0.0628163	0.0590605	0.0343173	3.6744210	-0.1045145	-0.3846549	NA	0.0279352	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	3467.119	14.704302	0.0006410	0.0006029	0.029548	2.8975115	NA	-0.3688099	NA	0.0225491	NA
preserved ~ stimlen + pos	3470.843	18.428855	0.0000996	0.0000936	0.030453	6.176883	-0.0880384	-0.1326825	NA	NA	NA
preserved ~ stimlen * pos	3473.412	20.998262	0.0000276	0.0000259	0.030610	2.5928195	-0.0584192	-0.2058996	-0.2008500	NA	NA
preserved ~ pos	3476.042	23.627735	0.0000074	0.0000070	0.026960	2.1604437	NA	-0.1612783	NA	NA	NA
preserved ~ stimlen	3510.415	37.999155	0.0000000	0.0000000	0.014925	3.8197909	-0.1603016	NA	NA	NA	NA
preserved ~ 1	3543.179	90.761267	0.0000000	0.0000000	0.0000000	1.940038	NA	NA	NA	NA	NA

```r
print(BestLPModelFormula)
```

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

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          1.40765           0.21412          -0.04578           0.62976           0.01065
##      stimlen:pos
##         -0.13915
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4396 Residual
## Null Deviance:       3094
## Residual Deviance: 3006  AIC: 3452
```

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

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

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

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

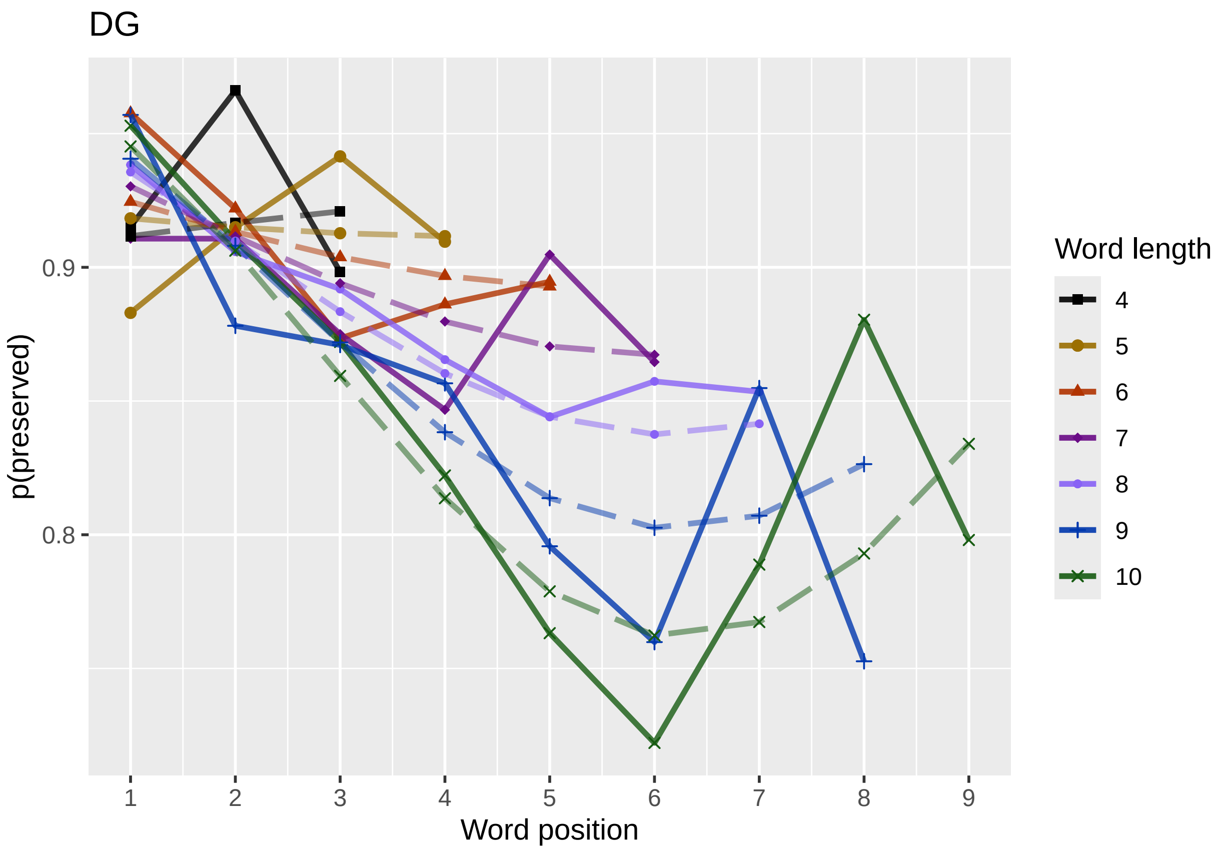
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.912 0.917 0.921 NA     NA     NA     NA     NA     NA
## 2       5 0.918 0.915 0.913  0.912 NA     NA     NA     NA     NA
## 3       6 0.925 0.913 0.904  0.897  0.893 NA     NA     NA     NA
## 4       7 0.930 0.912 0.894  0.880  0.870  0.867 NA     NA     NA
## 5       8 0.936 0.910 0.883  0.860  0.844  0.838  0.841 NA     NA
## 6       9 0.941 0.908 0.872  0.838  0.814  0.803  0.807  0.826 NA
## 7      10 0.945 0.906 0.859  0.814  0.779  0.762  0.767  0.793  0.834
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1        9   715
```

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

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

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

```
## [1] "The number of responses with fragments was 9 / 715 = 1.26 percent"
```

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

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

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

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

```
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##     (Intercept)          stimlen          I(pos^2)              pos  stimlen:I(pos^2)
##         1.27526          0.23546          -0.04802          0.70176           0.01162
##     stimlen:pos
##        -0.15186
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4378 Residual
## Null Deviance:      3019
## Residual Deviance: 2938  AIC: 3388
## log likelihood:  -1468.971
## Nagelkerke R2:  0.03675694
## % pres/err predicted correctly:  -895.5156
## % of predictable range [ (model-null)/(1-null) ]:  0.02349691
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.81557     -0.11119      0.03214     -0.40416
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4380 Residual
## Null Deviance:      3019
## Residual Deviance: 2951  AIC: 3395
## log likelihood:  -1475.683
## Nagelkerke R2:  0.03070851
## % pres/err predicted correctly:  -898.9261
## % of predictable range [ (model-null)/(1-null) ]:  0.01978212
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      2.9961       0.0264      -0.3871
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4381 Residual
## Null Deviance:      3019
## Residual Deviance: 2963  AIC: 3406
## log likelihood:  -1481.707
## Nagelkerke R2:  0.02526369
## % pres/err predicted correctly:  -902.7939
## % of predictable range [ (model-null)/(1-null) ]:  0.01556917
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos
##     3.17589     -0.09286     -0.11639
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4381 Residual
## Null Deviance:       3019
## Residual Deviance: 2962  AIC: 3411
## log likelihood:  -1481.182
## Nagelkerke R2:  0.02573896
## % pres/err predicted correctly:  -902.061
## % of predictable range [ (model-null)/(1-null) ]:  0.01636752
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    2.915701    -0.061861    -0.038382    -0.009004
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4380 Residual
## Null Deviance:       3019
## Residual Deviance: 2962  AIC: 3414
## log likelihood:  -1480.997
## Nagelkerke R2:  0.02590629
## % pres/err predicted correctly:  -902.007
## % of predictable range [ (model-null)/(1-null) ]:  0.01642632
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.5700      -0.1462
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4382 Residual
## Null Deviance:       3019
## Residual Deviance: 2971  AIC: 3417
## log likelihood:  -1485.515
## Nagelkerke R2:  0.02181469
## % pres/err predicted correctly:  -904.6712
## % of predictable range [ (model-null)/(1-null) ]:  0.01352437
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.1897      -0.1552
```

```
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4382 Residual
## Null Deviance:       3019
## Residual Deviance: 2988  AIC: 3442
## log likelihood:  -1494.242
## Nagelkerke R2:  0.01388645
## % pres/err predicted correctly:  -909.0226
## % of predictable range [ (model-null)/(1-null) ]:  0.008784772
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.973
##
## Degrees of Freedom: 4383 Total (i.e. Null);  4383 Residual
## Null Deviance:       3019
## Residual Deviance: 3019  AIC: 3472
## log likelihood:  -1509.445
## Nagelkerke R2:  -4.461154e-16
## % pres/err predicted correctly:  -917.0878
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	3388.175	0.000000	1.0000000	0.9734060	0.0367561	1.9275261	0.2354585	0.7017574	-0.1518640	-0.0480247	0.0116208
preserved ~ stimlen + I(pos^2) + pos	3395.387	7.212253	0.0271568	0.0264346	0.0307083	1.5815570	-0.1111856	-0.4041578	NA	0.0321408	NA
preserved ~ I(pos^2) + pos	3405.770	17.595062	0.0001501	0.0001470	0.0252637	1.7996113	NA	-0.3871365	NA	0.0263984	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	3411.292	23.116329	0.0000096	0.0000093	0.0257393	0.175893	-0.0928617	-0.1163915	NA	NA	NA
preserved ~ stimlen * pos	3413.892	25.716932	0.0000026	0.0000025	0.0259063	0.3915701	-0.0618614	-0.0383818	-0.0090043	NA	NA
preserved ~ pos	3417.172	28.999908	0.0000005	0.0000005	0.0218142	0.7569965	NA	-0.1461961	NA	NA	NA
preserved ~ stimlen	3441.675	53.500778	0.0000000	0.0000000	0.0138865	0.5189672	-0.1551687	NA	NA	NA	NA
preserved ~ 1	3471.618	83.441234	0.0000000	0.0000000	0.0000000	1.973124	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.
```

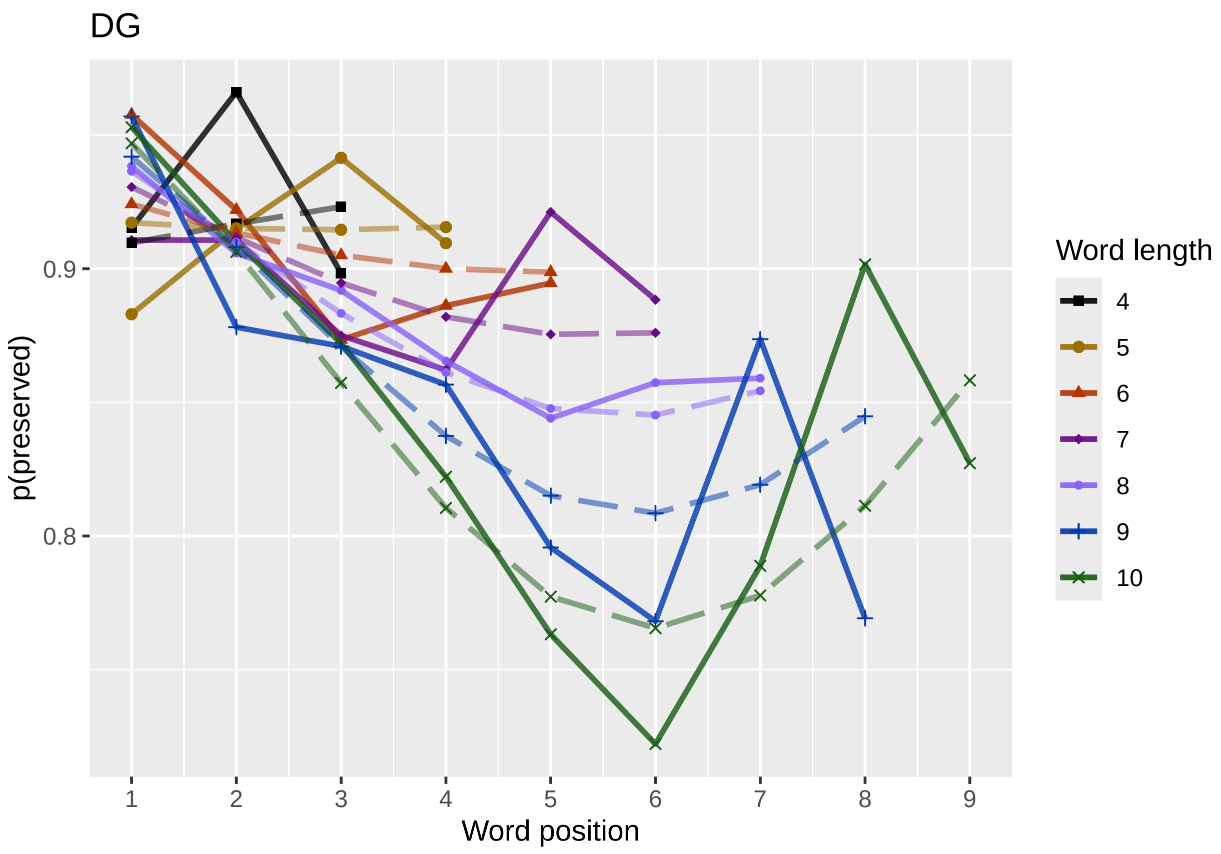
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.910 0.917 0.923 NA     NA     NA     NA     NA     NA
## 2       5 0.917 0.915 0.915  0.916 NA     NA     NA     NA     NA
## 3       6 0.924 0.913 0.905  0.900  0.899 NA     NA     NA     NA
## 4       7 0.931 0.912 0.895  0.882  0.875  0.876 NA     NA     NA
## 5       8 0.936 0.910 0.883  0.861  0.848  0.845  0.854 NA     NA
## 6       9 0.942 0.908 0.871  0.837  0.815  0.809  0.819  0.845 NA
## 7      10 0.947 0.906 0.857  0.810  0.777  0.766  0.778  0.811  0.858
```

```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: 0.70 - 0.99"
```

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

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

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

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

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.01168138
```

```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] -0.01855747
```

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

```r
if(potential_u_shape){ # downward, so potential U-shape
  # take only positive values from the table
  filtered_pos_upward_u<-table_pos_diffs
  filtered_pos_upward_u[table_pos_diffs <= 0] <- NA
  average_pos_u_diffs <- apply(filtered_pos_upward_u,
                               2,mean,na.rm=TRUE)
  OA_mean_pos_u_diff <- mean(average_pos_u_diffs,na.rm=TRUE)
  if(is.nan(OA_mean_pos_u_diff) | (OA_mean_pos_u_diff <= 0)){
    print("No U-shape in this participant")
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
    potential_u_shape <- FALSE
  }else{
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 1)

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

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

```
## [1] "Average upward change after U minimum"
## [1] 0.01544245
```

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

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

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

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

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

  print(" ")
  # percentage return upward
  percentage_return_upward <- upward_dist[biggest_return_upward_row]/downward_dist[biggest_return_upward
  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] 0.00000000 0.00666285 0.03167308 0.06298334 0.09809623 0.13797940 0.18287577
## [1] "differences from min to right max for each row: "
## [1] 0.009269166 0.000000000 0.000000000 0.000000000 0.003955098 0.023784836 0.071659734
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.1828758
## [1] 0.07165973
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.3918493
```

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

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

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

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

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

```
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          3.54307           -0.08498            0.03510            0.02468           -0.35774
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.02244           -0.01119            0.10585
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4394 Residual
## Null Deviance:      3094
## Residual Deviance: 3005  AIC: 3452
## log likelihood:  -1502.538
## Nagelkerke R2:  0.03964983
## % pres/err predicted correctly:  -920.2923
## % of predictable range [ (model-null)/(1-null) ]:  0.02518196
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           3.56826           -0.08519            0.02448           -0.35915           -0.13025
## I(pos^2):log_freq       pos:log_freq
##          -0.01212            0.10748
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4395 Residual
## Null Deviance:       3094
## Residual Deviance: 3007  AIC: 3452
## log likelihood:  -1503.401
## Nagelkerke R2:  0.03888866
## % pres/err predicted correctly:  -920.5377
## % of predictable range [ (model-null)/(1-null) ]:  0.02492224
## ***************************
## 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)
##          1.40765           0.21412          -0.04578           0.62976           0.01065
##      stimlen:pos
##         -0.13915
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4396 Residual
## Null Deviance:       3094
## Residual Deviance: 3006  AIC: 3452
## log likelihood:  -1502.856
## Nagelkerke R2:  0.03936943
## % pres/err predicted correctly:  -920.4609
## % of predictable range [ (model-null)/(1-null) ]:  0.02500353
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          3.56765          -0.08536           0.25811           0.02700          -0.37618
## stimlen:log_freq
##         -0.02550
##
```

```
## Degrees of Freedom: 4401 Total (i.e. Null);  4396 Residual
## Null Deviance:       3094
## Residual Deviance: 3010  AIC: 3453
## log likelihood:  -1505
## Nagelkerke R2:  0.03747738
## % pres/err predicted correctly:  -921.7721
## % of predictable range [ (model-null)/(1-null) ]:  0.02361611
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     3.62113     -0.08837      0.02775     -0.38303      0.05261
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4397 Residual
## Null Deviance:       3094
## Residual Deviance: 3013  AIC: 3455
## log likelihood:  -1506.358
## Nagelkerke R2:  0.03627783
## % pres/err predicted correctly:  -922.1927
## % of predictable range [ (model-null)/(1-null) ]:  0.02317108
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##          2.93686            0.01993           -0.34423           -0.10309           -0.01243
##     pos:log_freq
##          0.10645
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4396 Residual
## Null Deviance:       3094
## Residual Deviance: 3013  AIC: 3457
## log likelihood:  -1506.691
## Nagelkerke R2:  0.03598324
## % pres/err predicted correctly:  -922.7203
## % of predictable range [ (model-null)/(1-null) ]:  0.02261286
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.74421     -0.10451      0.02794     -0.38465
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 3017  AIC: 3458
## log likelihood:  -1508.575
## Nagelkerke R2:  0.0343176
## % pres/err predicted correctly:  -923.3998
## % of predictable range [ (model-null)/(1-null) ]:  0.0218939
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          3.01460          -0.06887           0.27327          -0.13284          -0.02731
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4397 Residual
## Null Deviance:       3094
## Residual Deviance: 3018  AIC: 3465
## log likelihood:  -1509.047
## Nagelkerke R2:  0.03390077
## % pres/err predicted correctly:  -924.1206
## % of predictable range [ (model-null)/(1-null) ]:  0.02113118
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.97511      0.02255     -0.36881
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3028  AIC: 3467
## log likelihood:  -1513.961
## Nagelkerke R2:  0.02954877
## % pres/err predicted correctly:  -926.9392
## % of predictable range [ (model-null)/(1-null) ]:  0.01814887
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         3.014732         -0.069306          0.273920         -0.132071         -0.028480
##     log_freq:pos
##         0.001978
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4396 Residual
## Null Deviance:       3094
## Residual Deviance: 3018  AIC: 3467
## log likelihood:  -1509.033
## Nagelkerke R2:  0.03391251
## % pres/err predicted correctly:  -924.1059
## % of predictable range [ (model-null)/(1-null) ]:  0.02114673
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     3.05618     -0.07176     -0.13285      0.05339
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 3021  AIC: 3468
## log likelihood:  -1510.647
## Nagelkerke R2:  0.03248472
## % pres/err predicted correctly:  -924.6819
## % of predictable range [ (model-null)/(1-null) ]:  0.02053725
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     3.047592     -0.069434     -0.135708      0.082962     -0.006686
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4397 Residual
## Null Deviance:       3094
## Residual Deviance: 3021  AIC: 3470
## log likelihood:  -1510.46
## Nagelkerke R2:  0.03265034
## % pres/err predicted correctly:  -924.67
## % of predictable range [ (model-null)/(1-null) ]:  0.02054987
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.58444     -0.15457      0.06661
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3026  AIC: 3470
## log likelihood:  -1513.106
## Nagelkerke R2:  0.03030653
## % pres/err predicted correctly:  -926.211
## % of predictable range [ (model-null)/(1-null) ]:  0.01891933
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.17688     -0.08804     -0.13268
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3026  AIC: 3471
## log likelihood:  -1512.934
## Nagelkerke R2:  0.0304596
## % pres/err predicted correctly:  -925.837
## % of predictable range [ (model-null)/(1-null) ]:  0.01931511
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     2.594304     -0.157713      0.107986     -0.009486
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 3025  AIC: 3472
## log likelihood:  -1512.729
## Nagelkerke R2:  0.03064097
## % pres/err predicted correctly:  -926.1241
## % of predictable range [ (model-null)/(1-null) ]:  0.01901125
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.92819     -0.05842     -0.05900     -0.00850
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 3026  AIC: 3473
## log likelihood:  -1512.763
## Nagelkerke R2:  0.03061047
## % pres/err predicted correctly:  -925.7679
## % of predictable range [ (model-null)/(1-null) ]:  0.01938816
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.6044      -0.1613
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3034  AIC: 3476
## log likelihood:  -1516.879
## Nagelkerke R2:  0.02696006
## % pres/err predicted correctly:  -928.3171
## % of predictable range [ (model-null)/(1-null) ]:  0.0166909
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          3.03701          -0.14139           0.27227          -0.02722
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 3053  AIC: 3505
## log likelihood:  -1526.527
## Nagelkerke R2:  0.01837645
## % pres/err predicted correctly:  -933.614
## % of predictable range [ (model-null)/(1-null) ]:  0.01108619
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     3.07837     -0.14426      0.05289
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3056  AIC: 3508
## log likelihood:  -1528.129
## Nagelkerke R2:  0.01694804
## % pres/err predicted correctly:  -934.0782
## % of predictable range [ (model-null)/(1-null) ]:  0.01059506
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.1979      -0.1603
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3061  AIC: 3510
## log likelihood:  -1530.394
## Nagelkerke R2:  0.01492581
## % pres/err predicted correctly:  -935.1915
## % of predictable range [ (model-null)/(1-null) ]:  0.00941706
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.94
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4401 Residual
## Null Deviance:       3094
## Residual Deviance: 3094  AIC: 3543
## log likelihood:  -1547.042
## Nagelkerke R2:  4.398286e-16
## % pres/err predicted correctly:  -944.0915
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	3451.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0849807	0.0350979	-0.0224013	-0.3577404	NA	0.10585[illegible]	0.0246819	NA	-0.0111865	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	3452.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.085[illegible]	-[illegible]02530	NA	-0.3591498	0.1074811	NA	0.0244763	-0.0121193	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	3452.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	0.6297562	NA	NA	-0.0457833	NA	NA	-0.1391492	0.0106536
preserved ~ stimlen * log_freq + I(pos^2) + pos	3453.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0853579	0.2581131	-0.0255013	-0.3761766	NA	NA	0.0270027	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	3455.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0883667	0.0526007	NA	-0.3830252	NA	NA	0.0277454	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	3457.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.1030913	NA	-0.3442336	0.1064453	NA	0.0199285	-0.0124273	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	3457.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1045145	NA	NA	-0.3846549	NA	NA	0.0279352	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	3465.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0688681	0.2732695	-0.0273018	-0.1328352	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	3467.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.3688099	NA	NA	0.0225491	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	3467.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0693061	0.2739202	-0.0284709	-0.1320708	NA	0.0019782	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	3468.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0717571	0.0533858	NA	-0.1328454	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)	I(pos^2):log_freq	[illegible]	[illegible]	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + pos * log_freq	3469.173962700641257004032835047592	-0.0694345	0.0829612	NA	-0.1357007	-0.0766856	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	3470.18437724500508390027503056544	NA	0.0666101	NA	-0.1545684	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	3470.8943172227550687002250451976883	-0.0880384	NA	NA	-0.1326825	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	3471.58910589004750015602459430	NA	0.1079857	NA	-0.1577130	-0.1094858	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	3473211374068101900006322628195	-0.0584192	NA	NA	-0.0589962	NA	NA	NA	NA	NA	-0.0085004	NA
preserved ~ pos	34762423701055005100027696014438	NA	NA	NA	-0.1612783	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	35053563840702800000001837637013	-0.1413876	0.2722703	-0.0272241	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	3507562405036240000000169478367	-0.1442620	0.0528938	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	3510548374257750000000149258909	-0.1603016	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	3543917604687700000000000940038	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##           3.54307           -0.08498            0.03510            0.02468           -0.35774
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##          -0.02244           -0.01119            0.10585
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4394 Residual
## Null Deviance:       3094
## Residual Deviance: 3005  AIC: 3452
```

```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"
```

```
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.
```

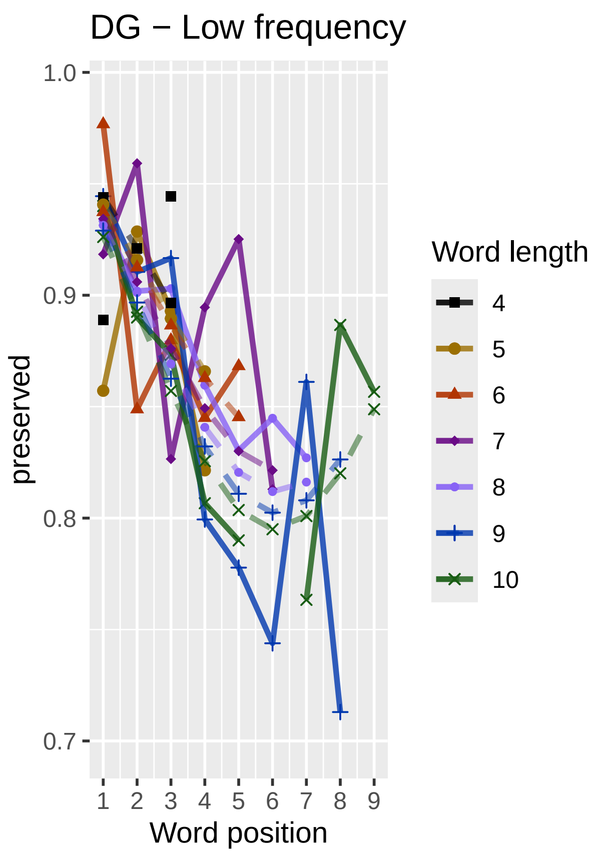
```
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

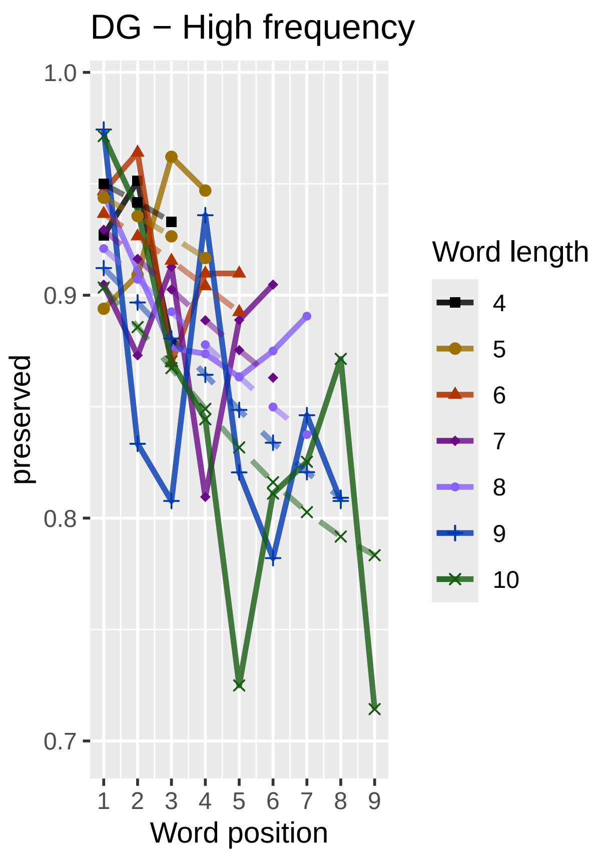
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale 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 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

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





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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.2748      -0.7354
```

```
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 2910  AIC: 3331
## log likelihood:  -1454.867
## Nagelkerke R2:  0.08124085
## % pres/err predicted correctly:  -883.0749
## % of predictable range [ (model-null)/(1-null) ]:  0.06456156
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.97511      0.02255     -0.36881
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3028  AIC: 3467
## log likelihood:  -1513.961
## Nagelkerke R2:  0.02954877
## % pres/err predicted correctly:  -926.9392
## % of predictable range [ (model-null)/(1-null) ]:  0.01814887
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.6044      -0.1613
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3034  AIC: 3476
## log likelihood:  -1516.879
## Nagelkerke R2:  0.02696006
## % pres/err predicted correctly:  -928.3171
## % of predictable range [ (model-null)/(1-null) ]:  0.0166909
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.1979      -0.1603
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3061  AIC: 3510
## log likelihood:  -1530.394
## Nagelkerke R2:  0.01492581
## % pres/err predicted correctly:  -935.1915
## % of predictable range [ (model-null)/(1-null) ]:  0.00941706
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.13596     -0.07467
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3084  AIC: 3533
## log likelihood:  -1541.86
## Nagelkerke R2:  0.004658406
## % pres/err predicted correctly:  -941.7696
## % of predictable range [ (model-null)/(1-null) ]:  0.002456763
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.94
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4401 Residual
## Null Deviance:       3094
## Residual Deviance: 3094  AIC: 3543
## log likelihood:  -1547.042
## Nagelkerke R2:  4.398286e-16
## % pres/err predicted correctly:  -944.0915
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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))
```

```r
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	3331.366	0.0000	1	1	0.0812408	3.274849	NA	-0.7354075	NA	NA	NA
preserved ~ (I(pos^2) + pos)	3467.119	135.7530	0	0	0.0295488	2.975115	NA	NA	0.0225491	-0.3688099	NA
preserved ~ pos	3476.042	144.6765	0	0	0.0269601	2.604437	NA	NA	NA	-0.1612783	NA
preserved ~ stimlen	3510.413	179.0479	0	0	0.0149258	3.197909	NA	NA	NA	NA	-0.1603016
preserved ~ CumPres	3533.394	202.0281	0	0	0.0046584	1.135961	-0.0746652	NA	NA	NA	NA
preserved ~ 1	3543.176	211.8100	0	0	0.0000000	0.940038	NA	NA	NA	NA	NA

```r
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	0.8839368	580
O	0.8547807	2029
P	1.0000000	36
S	0.7723357	256
V	0.9115026	1501

```
#  main effects models for data without satellite positions

keep_components = c("O","V","1")
OV1Data <- PosDat[PosDat$syll_component %in% keep_components,]
OV1Data <- OV1Data %>% select(stim_number,
                            stimlen,stim,pos,
                            preserved,syll_component)
OV1Data$CumPres <- CalcCumPres(OV1Data)
OV1Data$CumErr <- CalcCumErrFromPreserved(OV1Data)

SimpSyllMEAICSummary <- EvaluateSubsetData(OV1Data,MEModelEquations)
```

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.292       -0.750
##
## Degrees of Freedom: 4109 Total (i.e. Null);  4108 Residual
## Null Deviance:       2809
## Residual Deviance: 2655  AIC: 3037
## log likelihood:  -1327.405
## Nagelkerke R2:  0.07436671
## % pres/err predicted correctly:  -799.8493
## % of predictable range [ (model-null)/(1-null) ]:  0.05910231
## ***************************
## model index:  3
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.03232      0.02302     -0.37209
##
## Degrees of Freedom: 4109 Total (i.e. Null);  4107 Residual
## Null Deviance:       2809
## Residual Deviance: 2750  AIC: 3151
## log likelihood:  -1375.081
## Nagelkerke R2:  0.02870806
## % pres/err predicted correctly:  -835.2901
## % of predictable range [ (model-null)/(1-null) ]:  0.01746389
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.6515      -0.1598
##
## Degrees of Freedom: 4109 Total (i.e. Null);  4108 Residual
## Null Deviance:       2809
## Residual Deviance: 2756  AIC: 3160
## log likelihood:  -1377.827
## Nagelkerke R2:  0.0260459
## % pres/err predicted correctly:  -836.6182
## % of predictable range [ (model-null)/(1-null) ]:  0.01590353
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.0744      -0.1386
##
## Degrees of Freedom: 4109 Total (i.e. Null);  4108 Residual
## Null Deviance:       2809
## Residual Deviance: 2786  AIC: 3199
## log likelihood:  -1393.165
## Nagelkerke R2:  0.01111009
## % pres/err predicted correctly:  -844.1807
## % of predictable range [ (model-null)/(1-null) ]:  0.007018476
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      CumPres
##     2.15783     -0.06866
##
## Degrees of Freedom: 4109 Total (i.e. Null);  4108 Residual
## Null Deviance:       2809
## Residual Deviance: 2802  AIC: 3214
## log likelihood:  -1400.793
## Nagelkerke R2:  0.003640986
## % pres/err predicted correctly:  -848.4913
## % of predictable range [ (model-null)/(1-null) ]:  0.001954072
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.989
##
## Degrees of Freedom: 4109 Total (i.e. Null);  4109 Residual
## Null Deviance:       2809
## Residual Deviance: 2809  AIC: 3220
## log likelihood:  -1404.501
## Nagelkerke R2:  4.484567e-16
## % pres/err predicted correctly:  -850.1545
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	3037.221	0.0000	1	1	0.0743667	2.291538	NA	-0.7500458	NA	NA	NA
preserved ~ (I(pos^2) + pos)	3151.186	113.9649	0	0	0.0287083	1.032325	NA	NA	0.0230239	-0.3720852	NA
preserved ~ pos	3159.894	122.6727	0	0	0.0260452	2.651528	NA	NA	NA	-0.1597825	NA
preserved ~ stimlen	3198.900	161.6794	0	0	0.0111103	1.074419	NA	NA	NA	NA	-0.1386389
preserved ~ CumPres	3213.623	176.4021	0	0	0.0036410	2.157832	-0.0686614	NA	NA	NA	NA
preserved ~ 1	3220.186	182.9649	0	0	0.0000000	1.988746	NA	NA	NA	NA	NA

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

keep_components = c("O","V")
```

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

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

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

## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.2402      -0.7766
##
## Degrees of Freedom: 3529 Total (i.e. Null);  3528 Residual
## Null Deviance:       2437
## Residual Deviance: 2333  AIC: 2635
## log likelihood:  -1166.599
## Nagelkerke R2:  0.05808886
## % pres/err predicted correctly:  -699.3013
## % of predictable range [ (model-null)/(1-null) ]:  0.04856642
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.02768      0.02421     -0.38063
##
## Degrees of Freedom: 3529 Total (i.e. Null);  3527 Residual
## Null Deviance:       2437
## Residual Deviance: 2383  AIC: 2695
## log likelihood:  -1191.606
## Nagelkerke R2:  0.03029968
## % pres/err predicted correctly:  -721.6156
## % of predictable range [ (model-null)/(1-null) ]:  0.01825019
## ***************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.6388      -0.1586
##
## Degrees of Freedom: 3529 Total (i.e. Null);  3528 Residual
## Null Deviance:       2437
## Residual Deviance: 2389  AIC: 2704
## log likelihood:  -1194.443
## Nagelkerke R2:  0.02712176
## % pres/err predicted correctly:  -723.0048
## % of predictable range [ (model-null)/(1-null) ]:  0.0163627
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.9483      -0.1239
##
## Degrees of Freedom: 3529 Total (i.e. Null);  3528 Residual
## Null Deviance:       2437
## Residual Deviance: 2421  AIC: 2742
## log likelihood:  -1210.493
## Nagelkerke R2:  0.009048341
## % pres/err predicted correctly:  -730.7736
## % of predictable range [ (model-null)/(1-null) ]:  0.005808049
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.1932      -0.1011
##
## Degrees of Freedom: 3529 Total (i.e. Null);  3528 Residual
## Null Deviance:       2437
## Residual Deviance: 2426  AIC: 2747
## log likelihood:  -1213.136
## Nagelkerke R2:  0.00605665
## % pres/err predicted correctly:  -732.6482
## % of predictable range [ (model-null)/(1-null) ]:  0.00326111
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##       1.982
##
## Degrees of Freedom: 3529 Total (i.e. Null);  3529 Residual
## Null Deviance:       2437
## Residual Deviance: 2437  AIC: 2757
## log likelihood:  -1218.474
## Nagelkerke R2:  -4.453351e-16
## % pres/err predicted correctly:  -735.0486
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

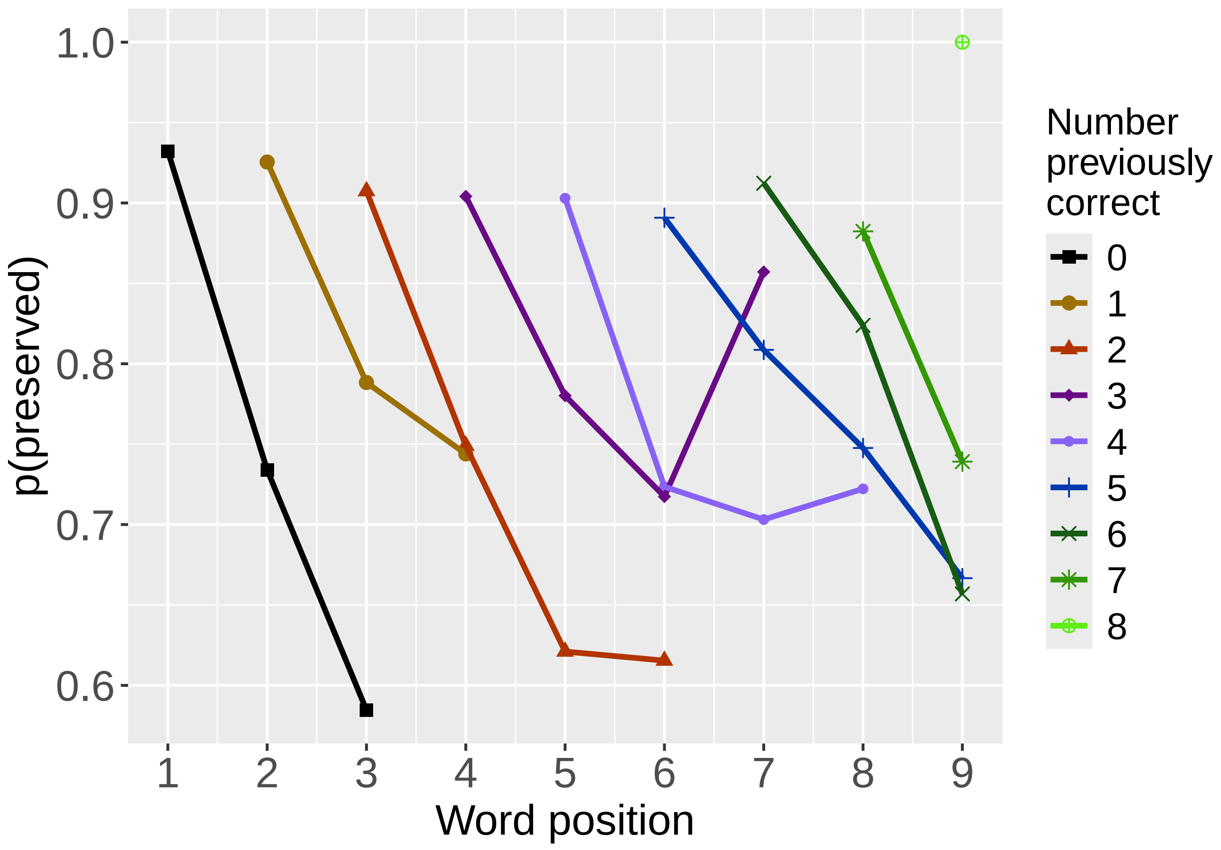
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2634.737	0.00000	1	1	0.0580889	2.240180	NA	-0.776645	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2694.940	60.20291	0	0	0.0302993	3.027678	NA	NA	0.0242093	-0.3806293	NA
preserved ~ pos	2703.579	68.84219	0	0	0.0271218	2.638823	NA	NA	NA	-0.1586442	NA
preserved ~ stimlen	2742.392	107.65533	0	0	0.0090483	2.948330	NA	NA	NA	NA	-0.1239263
preserved ~ CumPres	2746.541	111.80366	0	0	0.0060567	2.193192	-0.1010664	NA	NA	NA	NA
preserved ~ 1	2756.644	121.90741	0	0	0.0000000	1.982050	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.
```

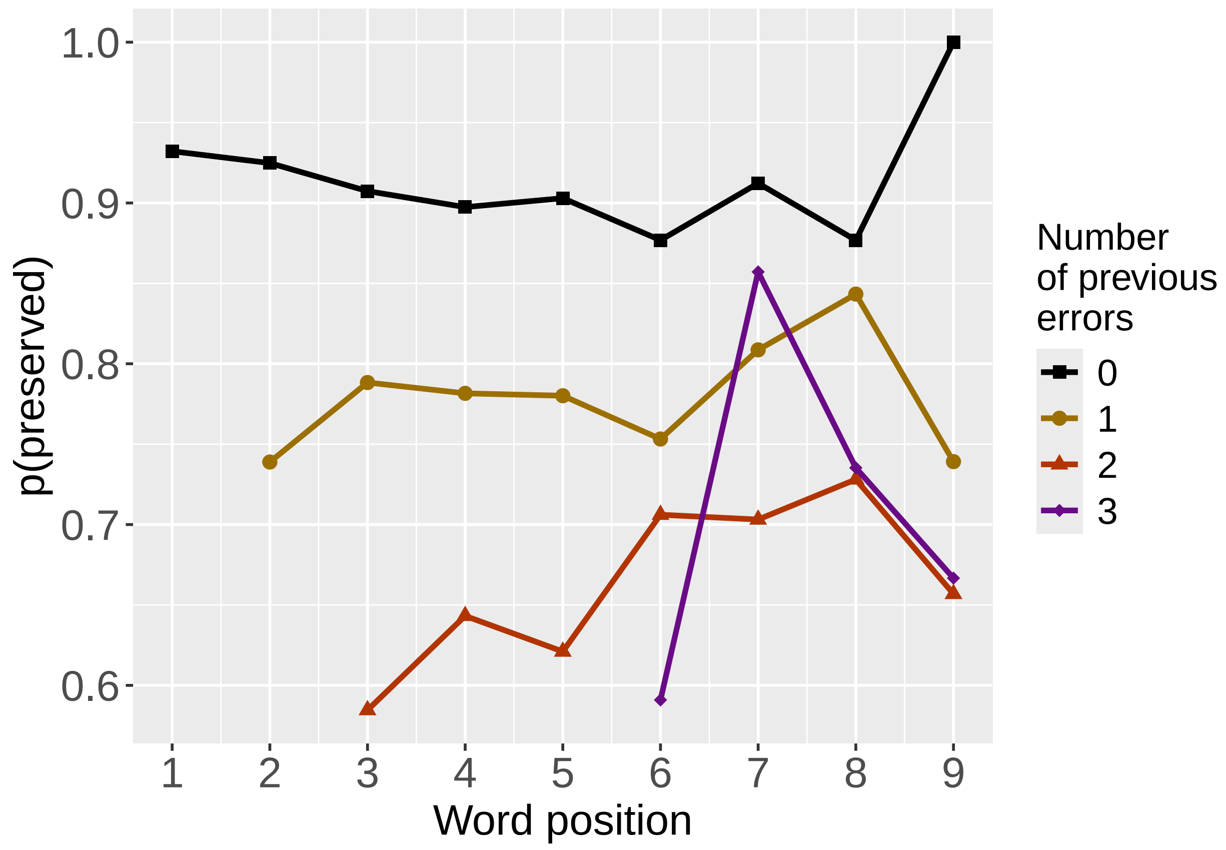
```
print(PrevCorPlot)
```



```
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
```

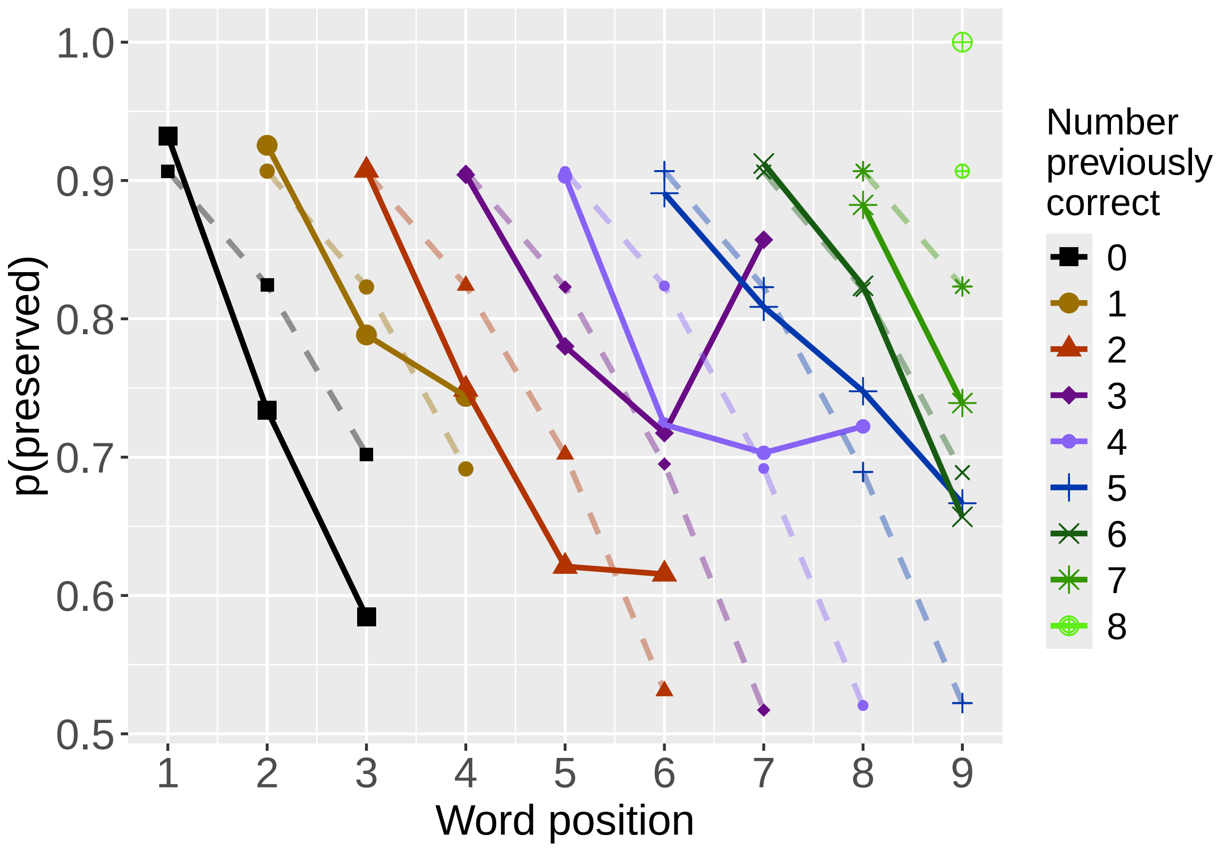
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

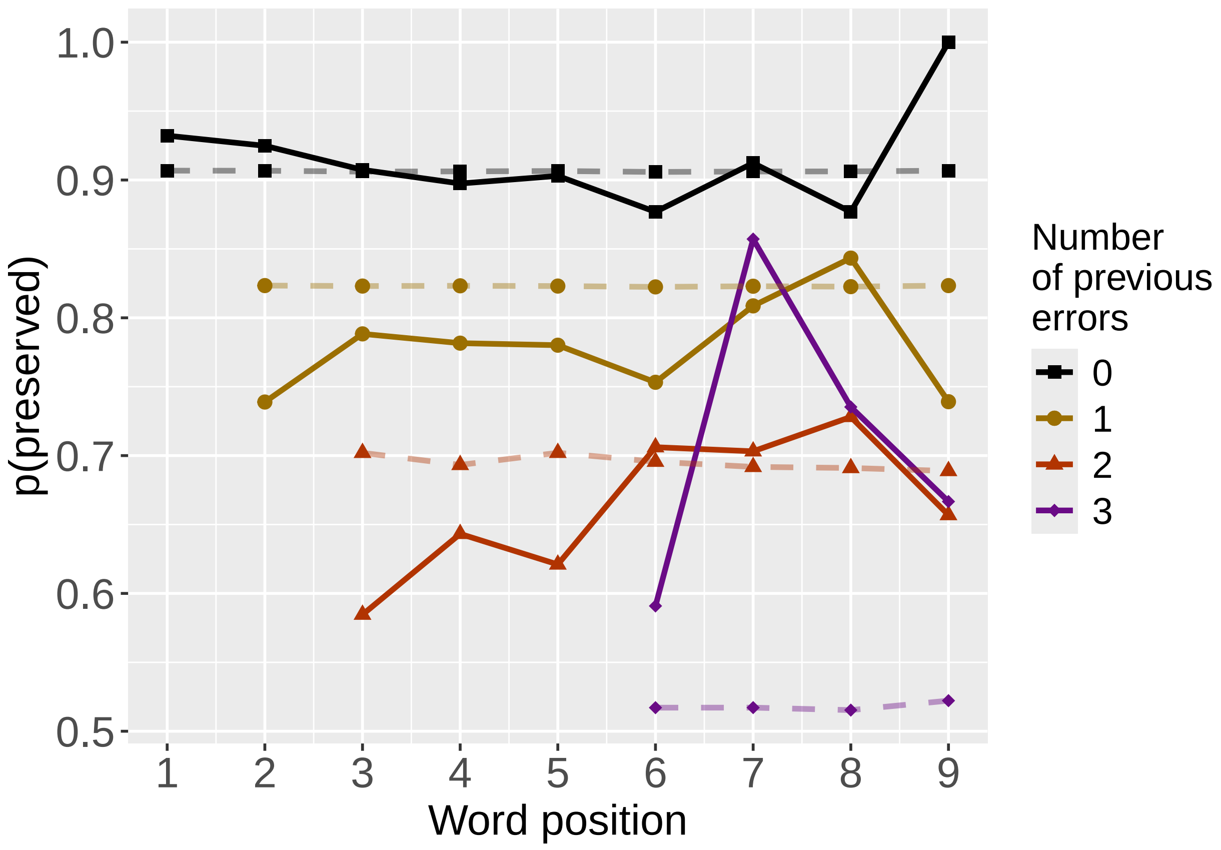
```
print(PrevCorPlot)
```



```r
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.
```

```r
print(PrevErrPlot)
```



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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.94722     -0.71583      0.03544     -0.35047
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 2894  AIC: 3311
## log likelihood:  -1447.242
## Nagelkerke R2:  0.08780994
## % pres/err predicted correctly:  -878.5578
## % of predictable range [ (model-null)/(1-null) ]:  0.06934108
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.2748      -0.7354
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 2910  AIC: 3331
## log likelihood:  -1454.867
## Nagelkerke R2:  0.08124085
## % pres/err predicted correctly:  -883.0749
## % of predictable range [ (model-null)/(1-null) ]:  0.06456156
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.97511      0.02255     -0.36881
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 3028  AIC: 3467
## log likelihood:  -1513.961
## Nagelkerke R2:  0.02954877
## % pres/err predicted correctly:  -926.9392
## % of predictable range [ (model-null)/(1-null) ]:  0.01814887
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	3311.427	0.00000	1.00e+00	0.9999532	0.0878099	2.947215	-0.7158269	0.0354406	-0.3504694

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	3331.366	19.93897	4.68e-05	0.0000468	0.0812408	2.274849	-0.7354075	NA	NA
preserved ~ I(pos^2) + pos	3467.119	155.69200	0.00e+00	0.0000000	0.0295488	2.975115	NA	0.0225491	-0.3688099

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     2.90535     -0.69751     -0.08266
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 2902  AIC: 3326
## log likelihood:  -1450.895
## Nagelkerke R2:  0.08466577
## % pres/err predicted correctly:  -881.7458
## % of predictable range [ (model-null)/(1-null) ]:  0.06596795
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.2748      -0.7354
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 2910  AIC: 3331
## log likelihood:  -1454.867
## Nagelkerke R2:  0.08124085
## % pres/err predicted correctly:  -883.0749
## % of predictable range [ (model-null)/(1-null) ]:  0.06456156
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.1979      -0.1603
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3061  AIC: 3510
## log likelihood:  -1530.394
## Nagelkerke R2:  0.01492581
## % pres/err predicted correctly:  -935.1915
## % of predictable range [ (model-null)/(1-null) ]:  0.00941706
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	3326.083	0.000000	1.000000	0.9334576	0.0846658	2.905347	-0.6975107	-0.0826648
preserved ~ CumErr	3331.366	5.282111	0.071286	0.0665424	0.0812408	2.274849	-0.7354075	NA
preserved ~ stimlen	3510.413	184.329995	0.000000	0.0000000	0.0149258	3.197909	NA	-0.1603016

```r
########
# 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
##      2.2748      -0.7354
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 2910  AIC: 3331
## log likelihood:  -1454.867
## Nagelkerke R2:  0.08124085
## % pres/err predicted correctly:  -883.0749
## % of predictable range [ (model-null)/(1-null) ]:  0.06456156
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     2.35667     -0.72414     -0.03297
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 2908  AIC: 3331
## log likelihood:  -1453.983
## Nagelkerke R2:  0.08200384
## % pres/err predicted correctly:  -883.0521
## % of predictable range [ (model-null)/(1-null) ]:  0.06458577
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.13596     -0.07467
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3084  AIC: 3533
## log likelihood:  -1541.86
## Nagelkerke R2:  0.004658406
## % pres/err predicted correctly:  -941.7696
## % of predictable range [ (model-null)/(1-null) ]:  0.002456763
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr	3331.366	0.0000000	1.0000000	0.5132313	0.0812408	2.274849	-0.7354075	NA
preserved ~ CumErr + CumPres	3331.471	0.1058753	0.9484391	0.4867687	0.0820038	2.356666	-0.7241379	-0.0329669
preserved ~ CumPres	3533.394	202.0281274	0.0000000	0.0000000	0.0046584	2.135961	NA	-0.0746652

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

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.2748      -0.7354
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 2910  AIC: 3331
## log likelihood:  -1454.867
## Nagelkerke R2:  0.08124085
## % pres/err predicted correctly:  -883.0749
## % of predictable range [ (model-null)/(1-null) ]:  0.06456156
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     2.38963     -0.69117     -0.03297
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 2908  AIC: 3331
## log likelihood:  -1453.983
## Nagelkerke R2:  0.08200384
## % pres/err predicted correctly:  -883.0521
## % of predictable range [ (model-null)/(1-null) ]:  0.06458577
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.6044      -0.1613
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 3034  AIC: 3476
## log likelihood:  -1516.879
## Nagelkerke R2:  0.02696006
## % pres/err predicted correctly:  -928.3171
## % of predictable range [ (model-null)/(1-null) ]:  0.0166909
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr	3331.366	0.0000000	1.0000000	0.5132313	0.0812408	2.274849	-0.7354075	NA
preserved ~ CumErr + pos	3331.471	0.1058753	0.9484391	0.4867687	0.0820038	2.389633	-0.6911710	-0.0329669
preserved ~ pos	3476.042	144.67646340	0.0000000	0.0000000	0.0269601	2.604437	NA	-0.1612783

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + I(pos^2) + pos	3311.427	0.0000000	1.0000000	0.9999531	0.0878092	2.947215	-0.7158269	0.0354406	-0.3504694	NA	NA
preserved ~ CumErr + stimlen	3326.083	0.0000000	1.0000000	0.9334576	0.0846652	2.905347	-0.6975107	NA	NA	-0.0826648	NA
preserved ~ CumErr	3331.366	19.9389714	0.0000468	0.0000468	0.0812408	2.274849	-0.7354075	NA	NA	NA	NA
preserved ~ CumErr	3331.366	5.2821114	0.0712860	0.0665424	0.0812408	2.274849	-0.7354075	NA	NA	NA	NA
preserved ~ CumErr	3331.366	0.0000000	1.0000000	0.5132313	0.0812408	2.274849	-0.7354075	NA	NA	NA	NA
preserved ~ CumErr	3331.366	0.0000000	1.0000000	0.5132313	0.0812408	2.274849	-0.7354075	NA	NA	NA	NA
preserved ~ CumErr + CumPres	3331.471	0.1058753	0.9484391	0.4867687	0.0820038	2.356666	-0.7241379	NA	NA	NA	-0.0329669
preserved ~ CumErr + pos	3331.471	0.1058753	0.9484391	0.4867687	0.0820038	2.389633	-0.6911710	NA	-0.0329669	NA	NA
preserved ~ I(pos^2) + pos	3467.119	155.6920017	0.0000000	0.0000000	0.0295488	2.975115	NA	0.0225491	-0.3688099	NA	NA
preserved ~ pos	3476.042	144.6764634	0.0000000	0.0000000	0.0269601	2.604437	NA	NA	-0.1612783	NA	NA
preserved ~ stimlen	3510.413	184.3299948	0.0000000	0.0000000	0.0149253	2.197909	NA	NA	NA	-0.1603016	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumPres	3533.394	202.02812	0.0000000	0.0000000	0.0046582	2.4135961	NA	NA	NA	NA	-0.0746652

```r
# explore influence of frequency and length

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

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

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

```
## ****************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     3.59784     -0.71285      0.04056     -0.36420     -0.08923      0.04434
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4396 Residual
## Null Deviance:       3094
## Residual Deviance: 2881  AIC: 3302
## log likelihood:  -1440.677
## Nagelkerke R2:  0.09344763
## % pres/err predicted correctly:  -875.2613
## % of predictable range [ (model-null)/(1-null) ]:  0.0728291
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     3.70001     -0.71633      0.04075     -0.36541     -0.10262
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4397 Residual
## Null Deviance:       3094
## Residual Deviance: 2884  AIC: 3303
## log likelihood:  -1442.183
## Nagelkerke R2:  0.09215596
## % pres/err predicted correctly:  -875.8665
## % of predictable range [ (model-null)/(1-null) ]:  0.07218884
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     2.94033     -0.71119      0.03614     -0.35123      0.06050
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4397 Residual
## Null Deviance:       3094
## Residual Deviance: 2889  AIC: 3307
## log likelihood:  -1444.277
## Nagelkerke R2:  0.09035887
## % pres/err predicted correctly:  -877.0665
## % of predictable range [ (model-null)/(1-null) ]:  0.07091911
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.94722     -0.71583      0.03544     -0.35047
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 2894  AIC: 3311
## log likelihood:  -1447.242
## Nagelkerke R2:  0.08780994
## % pres/err predicted correctly:  -878.5578
## % of predictable range [ (model-null)/(1-null) ]:  0.06934108
## **************************
## model index:  2
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.94
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4401 Residual
## Null Deviance:       3094
## Residual Deviance: 3094  AIC: 3543
## log likelihood:  -1547.042
## Nagelkerke R2:  4.398286e-16
## % pres/err predicted correctly:  -944.0915
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	3302.003	0.0000000	1.0000000	0.5926492	0.0934437	3.6597844	-0.7128517	0.0405615	-0.3641985	0.0443371	-0.0892317
preserved ~ CumErr + I(pos^2) + pos + stimlen	3302.980	0.9769701	0.6135552	0.3636230	0.0921563	3.6700012	-0.7163316	0.0407483	-0.3654094	NA	-0.1026172
preserved ~ CumErr + I(pos^2) + pos + log_freq	3307.476	5.4730623	0.0647947	0.0384005	0.0903582	3.5940326	-0.7111854	0.0361414	-0.3512274	0.0605004	NA
preserved ~ CumErr + I(pos^2) + pos	3311.427	9.4235229	0.0089889	0.0053273	0.0878092	3.5947215	-0.7158269	0.0354406	-0.3504694	NA	NA
preserved ~ 1	3543.176	241.1724895	0.0000000	0.0000000	0.0000000	1.9400380	NA	NA	NA	NA	NA

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

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

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

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

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

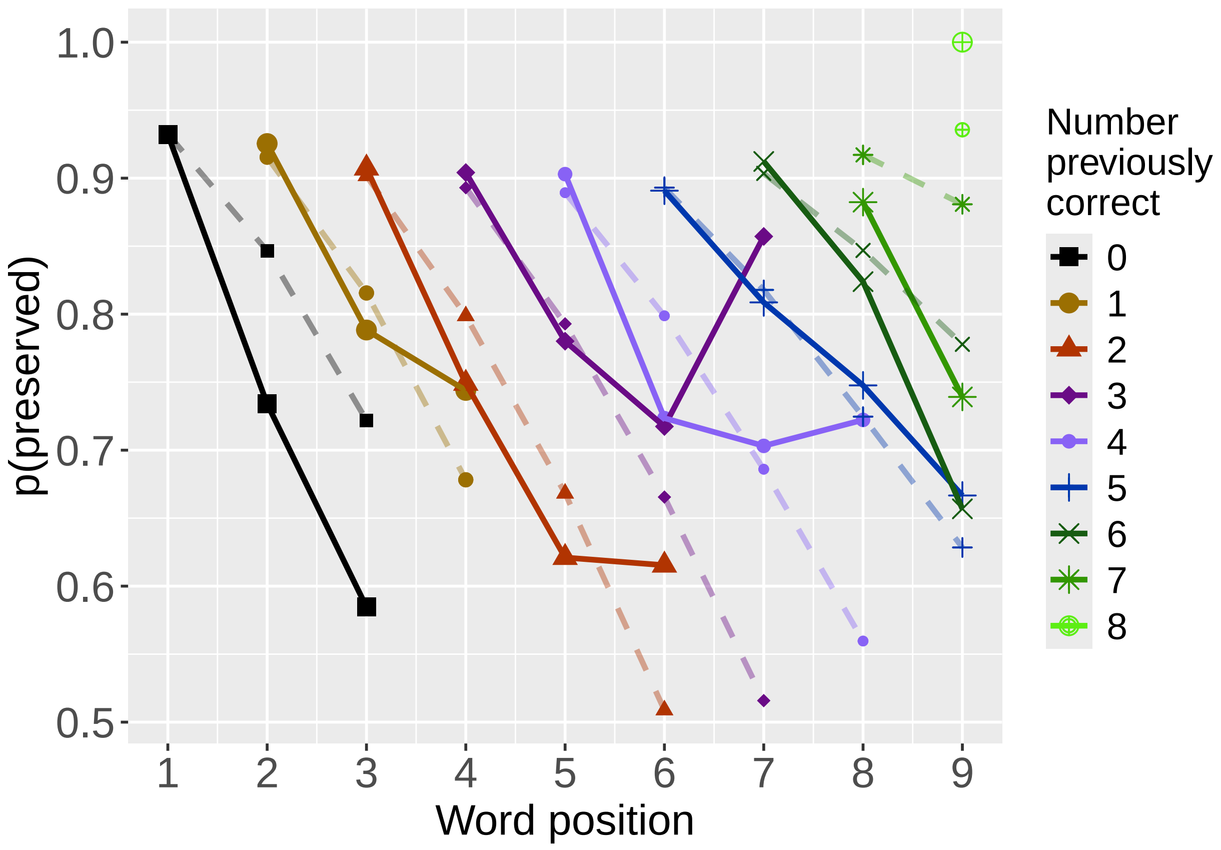
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   3012.7 3431.4
## I(pos^2)  1   2898.5 3317.1
## pos       1   2897.7 3316.3
## stimlen   1   2888.6 3307.2
## log_freq  1   2884.4 3303.0
## <none>        2881.3 3302.0
```

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

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

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

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

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

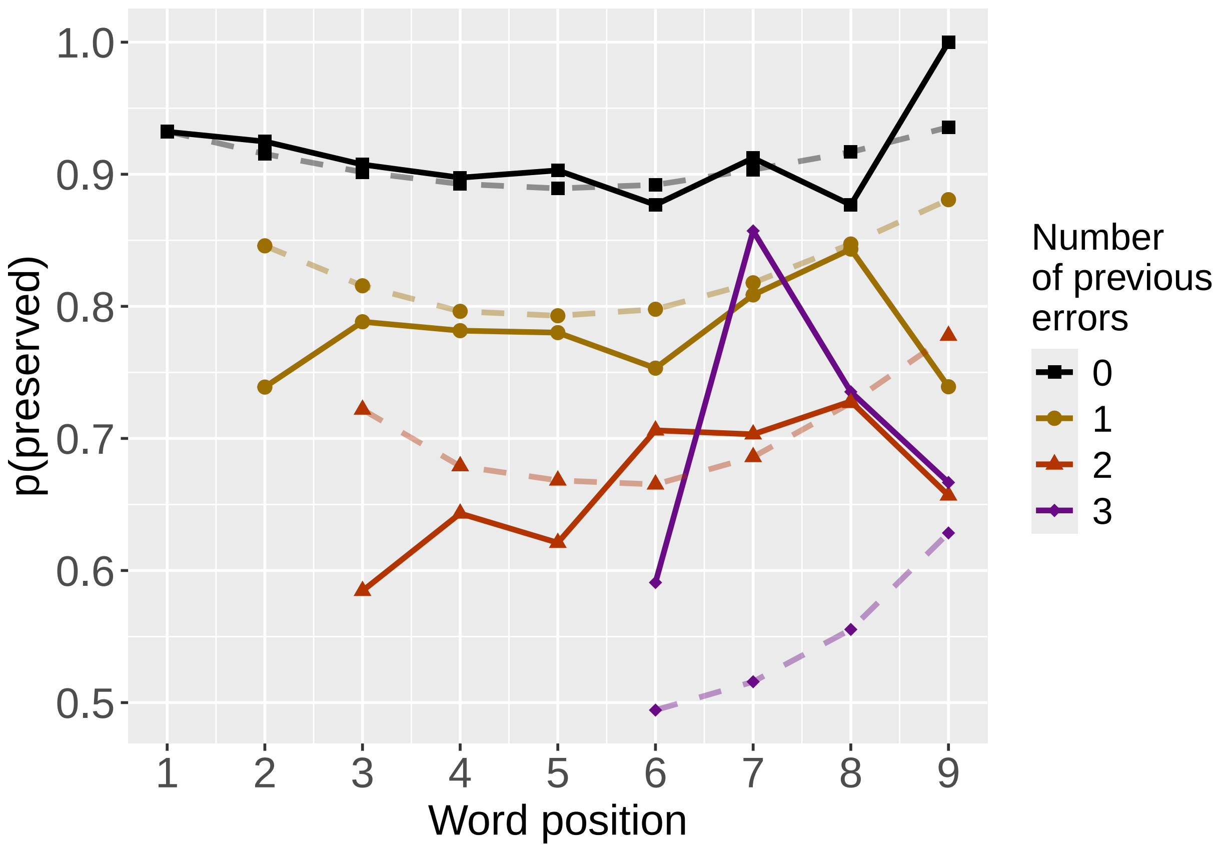
```
print(PrevCorPlot)
```



```r
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.
```

```r
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     I(pos^2)
##   2.2847952   -0.7272292   -0.0006433
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4399 Residual
## Null Deviance:       3094
## Residual Deviance: 2910  AIC: 3333
## log likelihood:  -1454.838
## Nagelkerke R2:  0.08126566
## % pres/err predicted correctly:  -883.1553
## % of predictable range [ (model-null)/(1-null) ]:  0.06447648
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.2748      -0.7354
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4400 Residual
## Null Deviance:       3094
## Residual Deviance: 2910  AIC: 3331
## log likelihood:  -1454.867
## Nagelkerke R2:  0.08124085
## % pres/err predicted correctly:  -883.0749
## % of predictable range [ (model-null)/(1-null) ]:  0.06456156
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.94722     -0.71583      0.03544     -0.35047
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4398 Residual
## Null Deviance:       3094
## Residual Deviance: 2894  AIC: 3311
## log likelihood:  -1447.242
## Nagelkerke R2:  0.08780994
## % pres/err predicted correctly:  -878.5578
## % of predictable range [ (model-null)/(1-null) ]:  0.06934108
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     3.70001     -0.71633      0.04075     -0.36541     -0.10262
##
## Degrees of Freedom: 4401 Total (i.e. Null);  4397 Residual
## Null Deviance:       3094
## Residual Deviance: 2884  AIC: 3303
## log likelihood:  -1442.183
## Nagelkerke R2:  0.09215596
## % pres/err predicted correctly:  -875.8665
## % of predictable range [ (model-null)/(1-null) ]:  0.07218884
## ***************************
## `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 6 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 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 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 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 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 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).
```

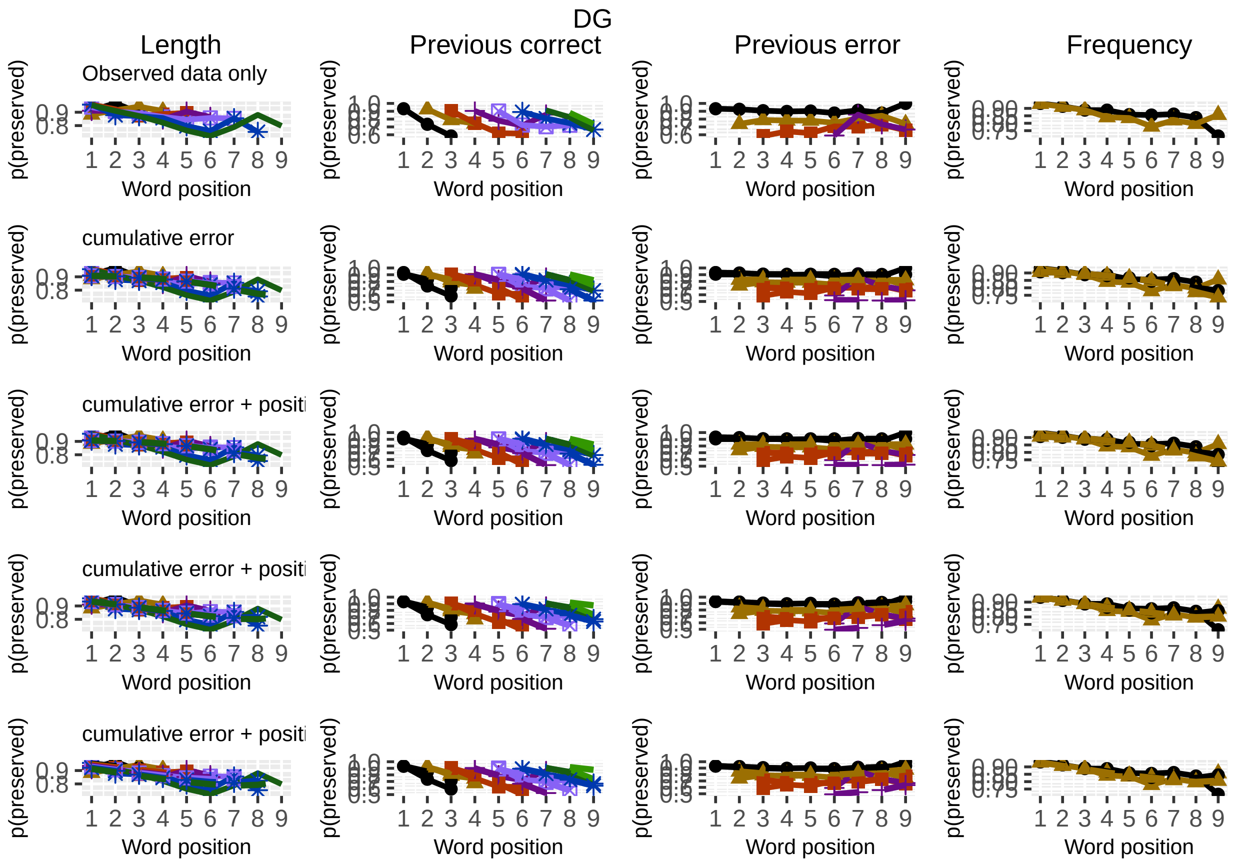
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## 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 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).
```

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



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

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

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

	CumErr	I(pos^2)	pos	stimlen	log_freq
McFadden	0.0483730	0.0073345	0.0095219	0.0038543	0.0018454
SquaredCorrelation	0.0377007	0.0057639	0.0074988	0.0030441	0.0014540
Nagelkerke	0.0377007	0.0057639	0.0074988	0.0030441	0.0014540
Estrella	0.0392224	0.0059352	0.0077010	0.0031151	0.0014924

```r
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!
```

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

```
##                                                               model deviance
## CumErr + I(pos^2) + pos + stimlen CumErr + I(pos^2) + pos + stimlen 2884.367
## CumErr + I(pos^2) + pos                     CumErr + I(pos^2) + pos 2894.484
## CumErr + I(pos^2)                                 CumErr + I(pos^2) 2909.676
## CumErr                                                       CumErr 2909.734
## null                                                           null 3094.084
##                                   deviance_explained percent_explained
## CumErr + I(pos^2) + pos + stimlen           209.7173          6.778011
## CumErr + I(pos^2) + pos                     199.5995          6.451004
## CumErr + I(pos^2)                           184.4076          5.960006
## CumErr                                      184.3501          5.958148
## null                                          0.0000          0.000000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + I(pos^2) + pos + stimlen                     100.00000             4.82452855
## CumErr + I(pos^2) + pos                                95.17547             7.24398098
## CumErr + I(pos^2)                                      87.93149             0.02741306
## CumErr                                                 87.90408            87.90407741
## null                                                         NA             0.00000000
```

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

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

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

	deviance	deviance_explained
CumErr + I(pos^2) + pos + stimlen	2884.367	209.7173
CumErr + I(pos^2) + pos	2894.484	199.5995
CumErr + I(pos^2)	2909.676	184.4076
CumErr	2909.734	184.3501
null	3094.084	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + I(pos^2) + pos + stimlen	6.778011	100.00000	4.8245285
CumErr + I(pos^2) + pos	6.451004	95.17547	7.2439810
CumErr + I(pos^2)	5.960006	87.93149	0.0274131
CumErr	5.958148	87.90408	87.9040774
null	0.000000	NA	0.0000000

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

```
##           Nagelkerke
## CumErr    0.67976397
## I(pos^2) 0.10392638
## pos       0.13520683
## stimlen   0.05488617
## log_freq 0.02621666
```

```r
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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

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

```r
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+I(pos^2)
## 1             preserved ~ CumErr+I(pos^2)       0.7072310       2909.676          0.000000000
## 2                      preserved ~ CumErr       0.7102513       2909.734          0.003020305
## 3         preserved ~ CumErr+I(pos^2)+pos       0.7270607       2894.484          0.019829724
## 4 preserved ~ CumErr+I(pos^2)+pos+stimlen       0.7304701       2884.367          0.023239144
##    diff_CumErr diff_CumErr+I(pos^2)+pos diff_CumErr+I(pos^2)+pos+stimlen
## 1 -0.003020305             -0.019829724                     -0.023239144
## 2  0.000000000             -0.016809419                     -0.020218839
## 3  0.016809419              0.000000000                     -0.003409419
## 4  0.020218839              0.003409419                      0.000000000
```

model	p_accounted_for	model_deviance
preserved ~ CumErr+I(pos^2)	0.7072310	2909.676
preserved ~ CumErr	0.7102513	2909.734
preserved ~ CumErr+I(pos^2)+pos	0.7270607	2894.484
preserved ~ CumErr+I(pos^2)+pos+stimlen	0.7304701	2884.367

model	diff_CumErr+I(pos^2)	diff_CumErr	diff_CumErr+I(pos^2)+pos
preserved ~ CumErr+I(pos^2)	0.0000000	-0.0030203	-0.0198297
preserved ~ CumErr	0.0030203	0.0000000	-0.0168094
preserved ~ CumErr+I(pos^2)+pos	0.0198297	0.0168094	0.0000000
preserved ~ CumErr+I(pos^2)+pos+stimlen	0.0232391	0.0202188	0.0034094

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

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

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