AV - repetition - Serial position analysis (v5)

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

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

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

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

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

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

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	537	33	130	NA	NA	700
2	64	NA	428	99	109	700
3	313	NA	168	204	15	700
4	297	NA	241	66	37	641
5	232	NA	205	72	38	547
6	202	1	139	69	22	433
7	175	NA	102	29	19	325
8	93	NA	55	25	4	177
9	75	NA	2	NA	7	84

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7671429	0.0471429	0.1857143	NA	NA	700
2	0.0914286	NA	0.6114286	0.1414286	0.1557143	700
3	0.4471429	NA	0.2400000	0.2914286	0.0214286	700
4	0.4633385	NA	0.3759750	0.1029641	0.0577223	641
5	0.4241316	NA	0.3747715	0.1316271	0.0694698	547
6	0.4665127	0.0023095	0.3210162	0.1593533	0.0508083	433

pos_factor	O	P	V	1	S	total
7	0.5384615	NA	0.3138462	0.0892308	0.0584615	325
8	0.5254237	NA	0.3107345	0.1412429	0.0225989	177
9	0.8928571	NA	0.0238095	NA	0.0833333	84

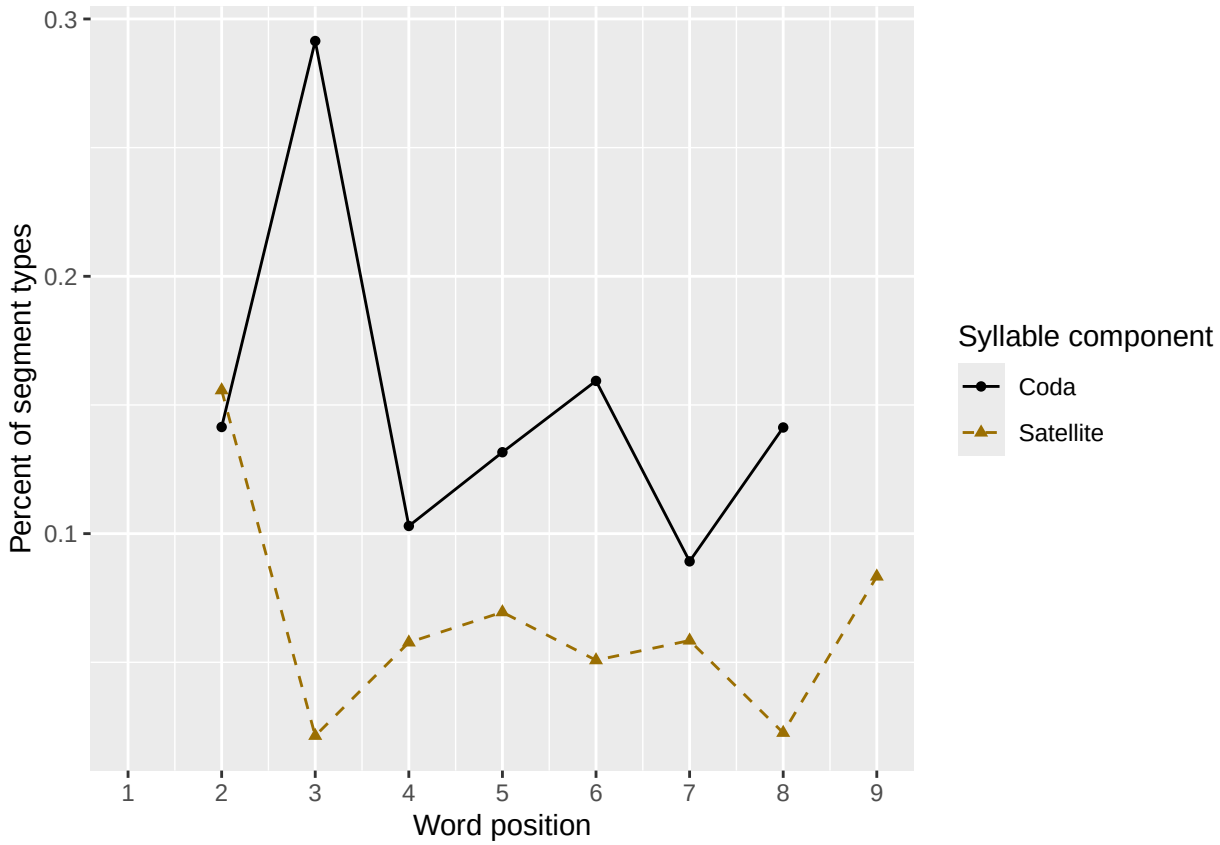
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.814 0.915 0.932 NA     NA     NA     NA     NA     NA
## 2       5 0.819 0.915 0.920  0.888 NA     NA     NA     NA     NA
## 3       6 0.734 0.842 0.947  0.947  0.880 NA     NA     NA     NA
## 4       7 0.685 0.864 0.881  0.870  0.958  0.926 NA     NA     NA
## 5       8 0.703 0.829 0.824  0.882  0.927  0.941  0.941 NA     NA
## 6       9 0.601 0.760 0.791  0.907  0.939  0.873  0.957  0.968 NA
## 7      10 0.637 0.768 0.875  0.946  0.929  0.917  0.929  0.964  0.869
```

```
# 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.
```

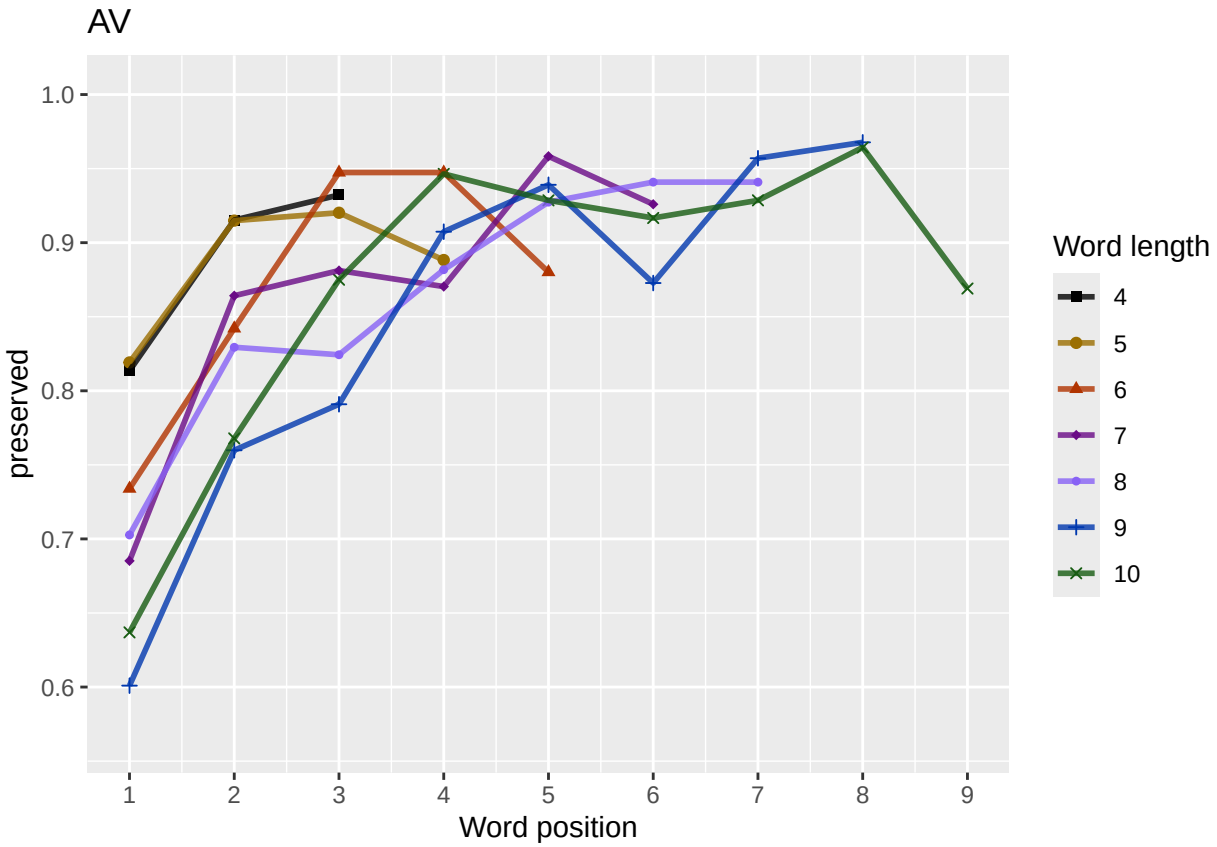
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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   114   114   114   114   114    NA    NA    NA    NA
## 4       7   108   108   108   108   108   108    NA    NA    NA
## 5       8   148   148   148   148   148   148   148    NA    NA
## 6       9    93    93    93    93    93    93    93    93    NA
## 7      10    84    84    84    84    84    84    84    84    84
```

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

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

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



Length and position

```
# length and position

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

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

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

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

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.0940034        -0.2586234        -0.0599673         0.4288493        -0.0007228
##      stimlen:pos
##        0.0497071
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 3118  AIC: 3213
## log likelihood:  -1559.074
## Nagelkerke R2:  0.09427551
## % pres/err predicted correctly:  -931.49
## % of predictable range [ (model-null)/(1-null) ]:  0.06134737
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.32735     -0.14564     -0.04979      0.72259
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3123  AIC: 3215
## log likelihood:  -1561.748
## Nagelkerke R2:  0.09209071
## % pres/err predicted correctly:  -933.8504
## % of predictable range [ (model-null)/(1-null) ]:  0.05897142
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      1.9607      -0.1603       0.3368
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3146  AIC: 3233
## log likelihood:  -1572.891
## Nagelkerke R2:  0.08295893
## % pres/err predicted correctly:  -941.0376
## % of predictable range [ (model-null)/(1-null) ]:  0.05173666
## ***************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    1.753126    -0.134958     0.420263    -0.009916
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3145  AIC: 3235
## log likelihood:  -1572.72
## Nagelkerke R2:  0.08309982
## % pres/err predicted correctly:  -941.1223
## % of predictable range [ (model-null)/(1-null) ]:  0.05165147
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.26061     -0.05683      0.74010
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3150  AIC: 3242
## log likelihood:  -1575.139
## Nagelkerke R2:  0.08111066
## % pres/err predicted correctly:  -943.0805
## % of predictable range [ (model-null)/(1-null) ]:  0.04968032
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.8628       0.2981
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3180  AIC: 3267
## log likelihood:  -1589.796
## Nagelkerke R2:  0.06901415
## % pres/err predicted correctly:  -952.1153
## % of predictable range [ (model-null)/(1-null) ]:  0.04058576
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##       1.868
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4306 Residual
## Null Deviance:       3343
## Residual Deviance: 3343  AIC: 3428
## log likelihood:  -1671.563
## Nagelkerke R2:  2.056531e-16
## % pres/err predicted correctly:  -992.4346
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.11555     -0.03204
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3342  AIC: 3429
## log likelihood:  -1670.829
## Nagelkerke R2:  0.000631358
## % pres/err predicted correctly:  -992.0361
## % of predictable range [ (model-null)/(1-null) ]:  0.0004011771
## ***************************
```

```
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)	3212.879	0.000000	1.000000	0.7338864	0.0942725	2.0940034	-0.2586234	0.4288403	0.0497071	-0.0599673	-0.0007228
preserved ~ stimlen + I(pos^2) + pos	3214.908	2.029200	0.362546	0.2660678	0.0920907	1.7273515	-0.1456352	0.7225853	NA	-0.0497886	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	3232.892	20.011680	0.0000045	0.0000038	0.1082958	1.9607048	-0.1602578	0.3368409	NA	NA	NA
preserved ~ stimlen * pos	3234.862	21.986496	0.0000016	0.0000014	0.1230831	0.7531257	-0.1349576	0.4202630	-0.0099164	NA	NA
preserved ~ I(pos^2) + pos	3241.812	28.936898	0.0000005	0.0000004	0.1811107	2.2606136	NA	0.7400956	NA	-0.0568268	NA
preserved ~ pos	3266.735	53.859814	0.0000000	0.0000000	0.0690141	1.8627698	NA	0.2980631	NA	NA	NA
preserved ~ 1	3428.347	215.467937	0.0000000	0.0000000	0.0000000	0.8683239	NA	NA	NA	NA	NA
preserved ~ stimlen	3428.502	215.625638	0.0000000	0.0000000	0.0006312	1.1155508	-0.0320397	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.0940034        -0.2586234        -0.0599673         0.4288493        -0.0007228
##      stimlen:pos
##        0.0497071
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 3118  AIC: 3213
```

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

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

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

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

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

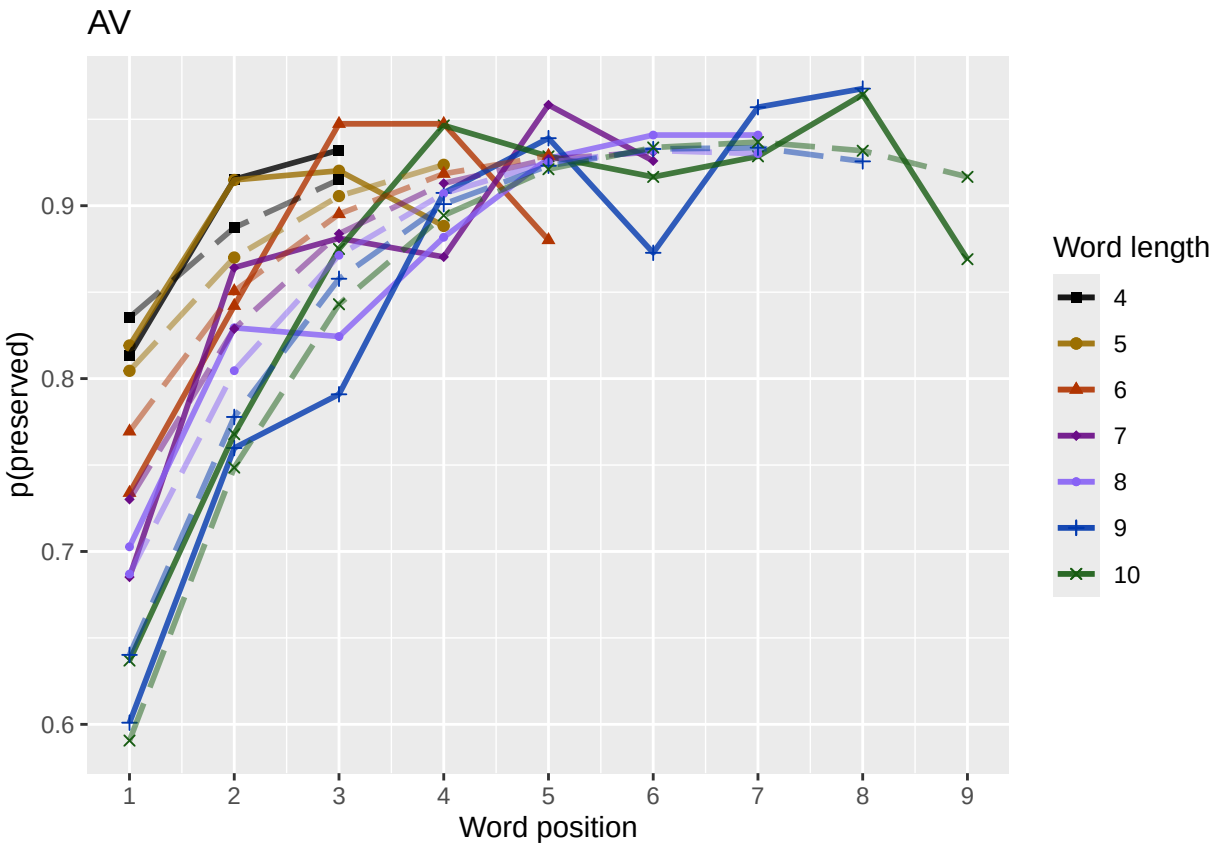
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.835 0.887 0.915 NA     NA     NA     NA     NA     NA
## 2       5 0.805 0.870 0.906  0.924 NA     NA     NA     NA     NA
## 3       6 0.769 0.851 0.895  0.918  0.929 NA     NA     NA     NA
## 4       7 0.730 0.829 0.884  0.913  0.927  0.931 NA     NA     NA
## 5       8 0.687 0.805 0.871  0.907  0.925  0.932  0.930 NA     NA
```

```
## 6       9 0.640 0.778 0.858  0.901  0.923  0.933  0.934  0.926 NA
## 7      10 0.591 0.749 0.843  0.894  0.921  0.934  0.937  0.932  0.917
```

```
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        5   700
```

```
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 5 / 700 = 0.71 percent"
```

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

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

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

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

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

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.0679623        -0.2563259        -0.0588099         0.4462514        -0.0005922
##      stimlen:pos
##        0.0475793
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4290 Residual
## Null Deviance:       3299
## Residual Deviance: 3059  AIC: 3154
## log likelihood:  -1529.624
## Nagelkerke R2:  0.1011384
## % pres/err predicted correctly:  -912.3527
## % of predictable range [ (model-null)/(1-null) ]:  0.0651018
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.33729     -0.14857     -0.04806      0.72671
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4292 Residual
## Null Deviance:       3299
## Residual Deviance: 3064  AIC: 3156
## log likelihood:  -1532.055
## Nagelkerke R2:  0.09914086
## % pres/err predicted correctly:  -914.5638
## % of predictable range [ (model-null)/(1-null) ]:  0.06283861
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      1.9306      -0.1618       0.3586
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4293 Residual
## Null Deviance:       3299
## Residual Deviance: 3084  AIC: 3171
## log likelihood:  -1541.79
## Nagelkerke R2:  0.09111592
## % pres/err predicted correctly:  -921.0445
## % of predictable range [ (model-null)/(1-null) ]:  0.05620498
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    1.765750    -0.141662     0.425834    -0.008015
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4292 Residual
## Null Deviance:       3299
## Residual Deviance: 3083  AIC: 3173
## log likelihood:  -1541.684
## Nagelkerke R2:  0.09120365
## % pres/err predicted correctly:  -921.1241
## % of predictable range [ (model-null)/(1-null) ]:  0.0561235
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.24887     -0.05522      0.74441
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4293 Residual
## Null Deviance:       3299
## Residual Deviance: 3092  AIC: 3184
## log likelihood:  -1545.834
## Nagelkerke R2:  0.08777238
## % pres/err predicted correctly:  -923.9953
## % of predictable range [ (model-null)/(1-null) ]:  0.05318461
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.8199       0.3201
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4294 Residual
## Null Deviance:       3299
## Residual Deviance: 3118  AIC: 3205
## log likelihood:  -1558.793
## Nagelkerke R2:  0.07701343
## % pres/err predicted correctly:  -932.2208
## % of predictable range [ (model-null)/(1-null) ]:  0.04476507
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.888
```

```
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4295 Residual
## Null Deviance:       3299
## Residual Deviance: 3299  AIC: 3384
## log likelihood:  -1649.341
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -975.9542
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.10792     -0.02857
##
## Degrees of Freedom: 4295 Total (i.e. Null);  4294 Residual
## Null Deviance:       3299
## Residual Deviance: 3298  AIC: 3385
## log likelihood:  -1648.766
## Nagelkerke R2:  0.0004989067
## % pres/err predicted correctly:  -975.6399
## % of predictable range [ (model-null)/(1-null) ]:  0.0003217328
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	3154.454	0.000000	1.000000	0.6813487	0.0101138	2.0679623	-0.2563259	0.4462510	0.0475793	-0.0588099	-0.0005922
preserved ~ stimlen + I(pos^2) + pos	3155.975	1.521238	0.4673769	0.3184466	0.0099140	1.9337293	-0.1485672	0.7267146	NA	-0.0480621	NA
preserved ~ stimlen + pos	3171.294	16.840142	0.0002204	0.0001502	0.0091115	1.9930570	-0.1617925	0.3585954	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	3173.331	18.878206	0.0000796	0.0000540	0.0912037	1.7657495	-0.1416620	0.4258341	-0.0080150	NA	NA
preserved ~ I(pos^2) + pos	3183.692	29.237124	0.0000004	0.0000003	0.0877792	2.2488725	NA	0.7444146	NA	-0.0552228	NA
preserved ~ pos	3205.359	50.904708	0.0000000	0.0000000	0.0770134	1.8199000	NA	0.3201477	NA	NA	NA
preserved ~ 1	3384.288	229.833897	0.0000000	0.0000000	0.0000000	1.8876058	NA	NA	NA	NA	NA
preserved ~ stimlen	3384.803	230.348539	0.0000000	0.0000000	0.0004982	2.1079243	-0.0285710	NA	NA	NA	NA

```
# 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.
```

```
# 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.
```

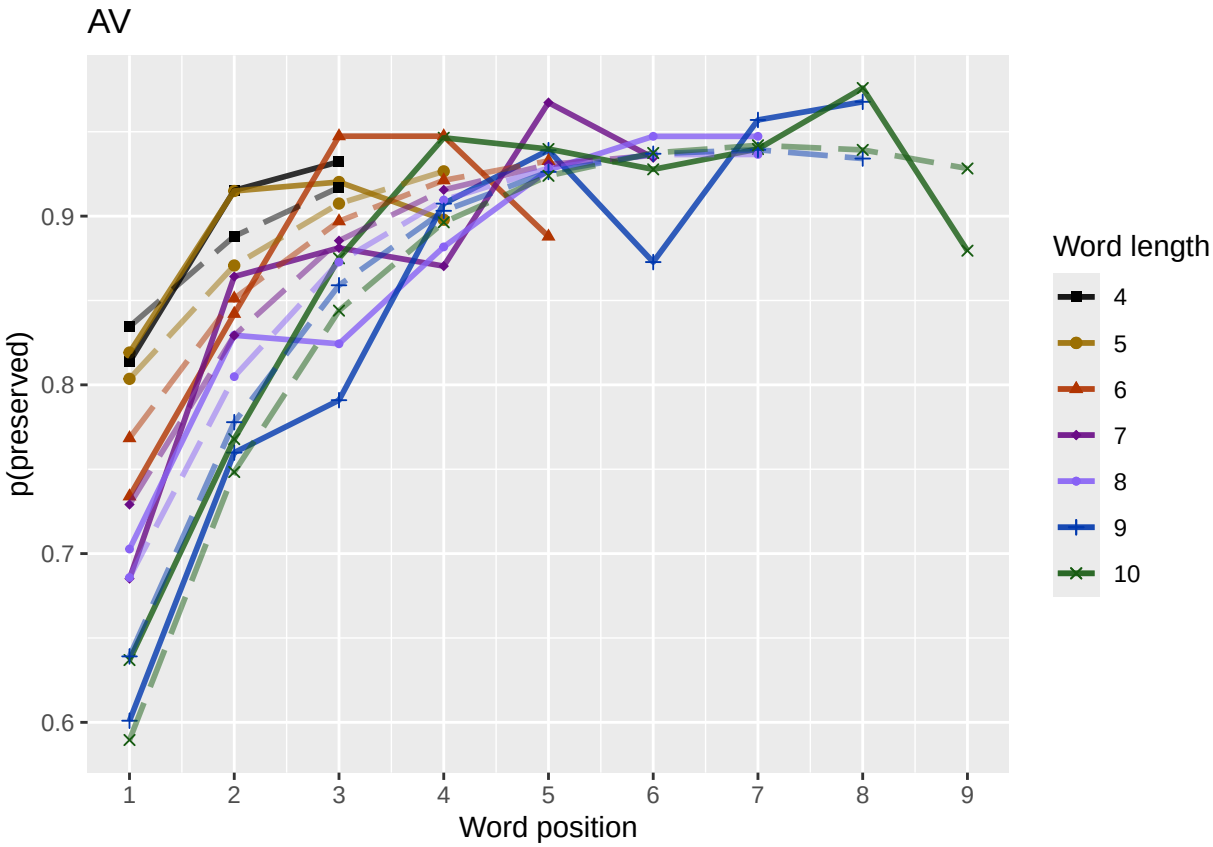
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.835 0.888 0.917 NA     NA     NA     NA     NA     NA
## 2       5 0.804 0.871 0.907  0.927 NA     NA     NA     NA     NA
## 3       6 0.768 0.851 0.897  0.921  0.933 NA     NA     NA     NA
## 4       7 0.729 0.829 0.885  0.916  0.931  0.936 NA     NA     NA
## 5       8 0.686 0.805 0.873  0.910  0.928  0.936  0.937 NA     NA
## 6       9 0.639 0.778 0.859  0.903  0.926  0.937  0.939  0.934 NA
## 7      10 0.590 0.748 0.844  0.896  0.924  0.937  0.942  0.939  0.928
```

```
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: 0.56 - 1.00"
```

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

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

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

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

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

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

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

```
print(OA_mean_len_diff)
```

```
## [1] -0.01527558
```

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

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

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

```
print(OA_mean_pos_diff)
```

```
## [1] 0.03687029
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         log_freq         I(pos^2)              pos
##         0.99916         -0.11161          0.60308         -0.05218          0.74253
## stimlen:log_freq
##        -0.06566
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 3090  AIC: 3183
## log likelihood:  -1545.014
## Nagelkerke R2:  0.1057163
## % pres/err predicted correctly:  -925.9795
## % of predictable range [ (model-null)/(1-null) ]:  0.06689428
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         1.003921         -0.114456          0.551648         -0.052584          0.750952
##  stimlen:log_freq  log_freq:I(pos^2)      log_freq:pos
##        -0.070132         -0.003642          0.044530
##
```

```
## Degrees of Freedom: 4306 Total (i.e. Null);  4299 Residual
## Null Deviance:       3343
## Residual Deviance: 3088  AIC: 3185
## log likelihood:  -1544.072
## Nagelkerke R2:  0.1064797
## % pres/err predicted correctly:  -925.9438
## % of predictable range [ (model-null)/(1-null) ]:  0.06693028
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          1.67030          -0.13090           0.56640           0.34760          -0.06965
##     log_freq:pos
##          0.02183
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 3112  AIC: 3202
## log likelihood:  -1555.776
## Nagelkerke R2:  0.09696562
## % pres/err predicted correctly:  -933.2446
## % of predictable range [ (model-null)/(1-null) ]:  0.05958117
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     1.09872     -0.11582     -0.05031      0.72778      0.09219
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4302 Residual
## Null Deviance:       3343
## Residual Deviance: 3110  AIC: 3203
## log likelihood:  -1555.05
## Nagelkerke R2:  0.09755712
## % pres/err predicted correctly:  -931.0606
## % of predictable range [ (model-null)/(1-null) ]:  0.06177962
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          1.67326          -0.12825           0.58474           0.33876          -0.06363
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4302 Residual
```

```
## Null Deviance:       3343
## Residual Deviance: 3114  AIC: 3203
## log likelihood:  -1557.017
## Nagelkerke R2:  0.09595385
## % pres/err predicted correctly:  -933.5475
## % of predictable range [ (model-null)/(1-null) ]:  0.05927632
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)               pos           log_freq
##          1.068764          -0.114179          -0.052951           0.744846           0.033079
## I(pos^2):log_freq       pos:log_freq
##         -0.006192           0.046914
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4300 Residual
## Null Deviance:       3343
## Residual Deviance: 3109  AIC: 3206
## log likelihood:  -1554.344
## Nagelkerke R2:  0.0981322
## % pres/err predicted correctly:  -931.0164
## % of predictable range [ (model-null)/(1-null) ]:  0.0618241
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.0940034        -0.2586234        -0.0599673         0.4288493         -0.0007228
##      stimlen:pos
##        0.0497071
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 3118  AIC: 3213
## log likelihood:  -1559.074
## Nagelkerke R2:  0.09427551
## % pres/err predicted correctly:  -931.49
## % of predictable range [ (model-null)/(1-null) ]:  0.06134737
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.32735     -0.14564     -0.04979      0.72259
##
```

```
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3123  AIC: 3215
## log likelihood:  -1561.748
## Nagelkerke R2:  0.09209071
## % pres/err predicted correctly:  -933.8504
## % of predictable range [ (model-null)/(1-null) ]:  0.05897142
## ***************************
## 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
##          0.226107         -0.058757          0.761966          0.066723          -0.006814
##      pos:log_freq
##          0.047503
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 3124  AIC: 3220
## log likelihood:  -1561.754
## Nagelkerke R2:  0.09208564
## % pres/err predicted correctly:  -936.0483
## % of predictable range [ (model-null)/(1-null) ]:  0.05675897
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     1.74161     -0.13102      0.33805      0.09064
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3133  AIC: 3221
## log likelihood:  -1566.355
## Nagelkerke R2:  0.08832075
## % pres/err predicted correctly:  -938.2742
## % of predictable range [ (model-null)/(1-null) ]:  0.05451832
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     1.741952     -0.131805      0.340657      0.074213      0.005522
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4302 Residual
```

```
## Null Deviance:       3343
## Residual Deviance: 3133  AIC: 3223
## log likelihood:  -1566.267
## Nagelkerke R2:  0.08839312
## % pres/err predicted correctly:  -938.3245
## % of predictable range [ (model-null)/(1-null) ]:  0.05446767
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      1.9607      -0.1603       0.3368
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3146  AIC: 3233
## log likelihood:  -1572.891
## Nagelkerke R2:  0.08295893
## % pres/err predicted correctly:  -941.0376
## % of predictable range [ (model-null)/(1-null) ]:  0.05173666
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    1.753126    -0.134958     0.420263    -0.009916
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3145  AIC: 3235
## log likelihood:  -1572.72
## Nagelkerke R2:  0.08309982
## % pres/err predicted correctly:  -941.1223
## % of predictable range [ (model-null)/(1-null) ]:  0.05165147
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      0.8347       0.3102       0.1230
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3153  AIC: 3241
## log likelihood:  -1576.608
```

```
## Nagelkerke R2:  0.07990197
## % pres/err predicted correctly:  -944.834
## % of predictable range [ (model-null)/(1-null) ]:  0.04791515
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.26061     -0.05683      0.74010
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3150  AIC: 3242
## log likelihood:  -1575.139
## Nagelkerke R2:  0.08111066
## % pres/err predicted correctly:  -943.0805
## % of predictable range [ (model-null)/(1-null) ]:  0.04968032
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     0.832935      0.311033      0.117874      0.001745
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3153  AIC: 3243
## log likelihood:  -1576.6
## Nagelkerke R2:  0.07990912
## % pres/err predicted correctly:  -944.8653
## % of predictable range [ (model-null)/(1-null) ]:  0.04788371
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.8628       0.2981
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3180  AIC: 3267
## log likelihood:  -1589.796
## Nagelkerke R2:  0.06901415
## % pres/err predicted correctly:  -952.1153
## % of predictable range [ (model-null)/(1-null) ]:  0.04058576
```

```
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         1.843234         -0.001037          0.567821         -0.061767
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3311  AIC: 3400
## log likelihood:  -1655.497
## Nagelkerke R2:  0.0137679
## % pres/err predicted correctly:  -985.1139
## % of predictable range [ (model-null)/(1-null) ]:  0.007369109
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##    1.907602    -0.003787     0.086133
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3329  AIC: 3418
## log likelihood:  -1664.63
## Nagelkerke R2:  0.005953698
## % pres/err predicted correctly:  -989.1624
## % of predictable range [ (model-null)/(1-null) ]:  0.003293852
## **************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.868
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4306 Residual
## Null Deviance:       3343
## Residual Deviance: 3343  AIC: 3428
## log likelihood:  -1671.563
## Nagelkerke R2:  2.056531e-16
## % pres/err predicted correctly:  -992.4346
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
## model index:  15
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.11555     -0.03204
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3342  AIC: 3429
## log likelihood:  -1670.829
## Nagelkerke R2:  0.000631358
## % pres/err predicted correctly:  -992.0361
## % of predictable range [ (model-null)/(1-null) ]:  0.0004011771
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	[illegible]	log_freq:pos	I(pos^2)	[illegible]	log_freq:I(pos^2)	[illegible]	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + I(pos^2) + pos	3183.4740	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1116147	0.6030772	-0.0656624	0.7425273	NA	NA	-0.0521750	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	3185.2872	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1144558	0.5516485	-0.0701317	0.7509518	NA	0.0445302	-0.0525842	NA	-0.0036418	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	3202.3032	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1309044	0.5663979	-0.0696506	0.3475981	NA	0.0218318	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos		I(pos^2)	log_freq:I(pos^2)		stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1158246	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0503077	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1282456	0.5847371	-0.0636346	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1141789	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2586234	NA	NA	[illegible]	[illegible]	NA	-0.0599673	NA	NA	0.0497071	-0.0007228
preserved ~ stimlen + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1456352	NA	NA	[illegible]	[illegible]	NA	-0.0497886	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1310156	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1318047	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1602578	NA	NA	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1349576	NA	NA	[illegible]	[illegible]	NA	NA	NA	NA	-0.0099164	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	[illegible]	NA	-0.0568268	NA	NA	NA	NA
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA
preserved ~ pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0010368	0.5678208	-0.0617674	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAICexpAICwNagR2(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + log_freq	3417.2314.1430800000000591537 6046	-0.0037873	0.0861330	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	3428.2447.8728500000000001868329	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	3428.2505.0305510000000231145508	-0.0320397	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          0.99916          -0.11161           0.60308          -0.05218           0.74253
## stimlen:log_freq
##         -0.06566
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 3090  AIC: 3183
```

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

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

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

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

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

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

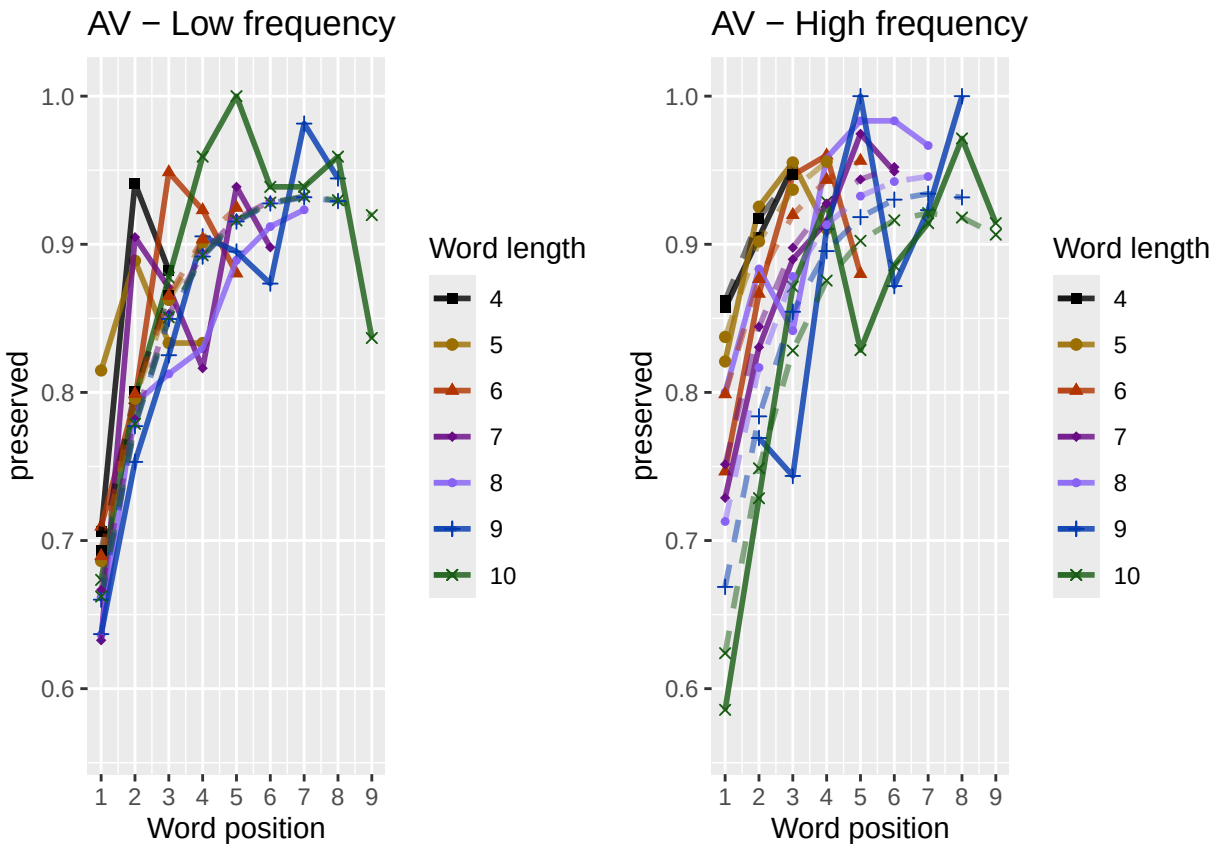
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

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

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

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




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

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.0830       0.4485
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3081  AIC: 3161
## log likelihood:  -1540.534
## Nagelkerke R2:  0.1093464
## % pres/err predicted correctly:  -928.473
## % of predictable range [ (model-null)/(1-null) ]:  0.06438434
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.26061     -0.05683      0.74010
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3150  AIC: 3242
## log likelihood:  -1575.139
## Nagelkerke R2:  0.08111066
## % pres/err predicted correctly:  -943.0805
## % of predictable range [ (model-null)/(1-null) ]:  0.04968032
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.8628       0.2981
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3180  AIC: 3267
## log likelihood:  -1589.796
## Nagelkerke R2:  0.06901415
## % pres/err predicted correctly:  -952.1153
## % of predictable range [ (model-null)/(1-null) ]:  0.04058576
```

```
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.9791      -0.1917
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3328  AIC: 3412
## log likelihood:  -1664.158
## Nagelkerke R2:  0.006359056
## % pres/err predicted correctly:  -988.0855
## % of predictable range [ (model-null)/(1-null) ]:  0.004377825
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.868
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4306 Residual
## Null Deviance:       3343
## Residual Deviance: 3343  AIC: 3428
## log likelihood:  -1671.563
## Nagelkerke R2:  2.056531e-16
## % pres/err predicted correctly:  -992.4346
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.11555     -0.03204
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3342  AIC: 3429
## log likelihood:  -1670.829
## Nagelkerke R2:  0.000631358
## % pres/err predicted correctly:  -992.0361
## % of predictable range [ (model-null)/(1-null) ]:  0.0004011771
## **************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	3160.708	0.00000	1	1	0.1093464	1.0830050	0.44848	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	3241.816	81.10798	0	0	0.0811107	0.2606136	NA	NA	-0.0568268	0.7400956	NA
preserved ~ pos	3266.739	106.03089	0	0	0.0690140	0.8627698	NA	NA	NA	0.2980631	NA
preserved ~ CumErr	3412.172	251.46334	0	0	0.0063591	1.9790594	NA	-0.191714	NA	NA	NA
preserved ~ 1	3428.347	267.63902	0	0	0.0000000	1.8683239	NA	NA	NA	NA	NA
preserved ~ stimlen	3428.505	267.79672	0	0	0.0006312	2.1155508	NA	NA	NA	NA	-0.0320397

```
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	0.9091312	564
O	0.8458110	1988
P	0.7352941	34
S	0.5830013	251
V	0.9288738	1470

```
#  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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
```

```
##      1.2458       0.4868
##
## Degrees of Freedom: 4021 Total (i.e. Null);  4020 Residual
## Null Deviance:       2825
## Residual Deviance: 2596  AIC: 2677
## log likelihood:  -1297.964
## Nagelkerke R2:  0.1097443
## % pres/err predicted correctly:  -763.0097
## % of predictable range [ (model-null)/(1-null) ]:  0.06204827
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      0.2103      -0.0743       0.8935
##
## Degrees of Freedom: 4021 Total (i.e. Null);  4019 Residual
## Null Deviance:       2825
## Residual Deviance: 2633  AIC: 2726
## log likelihood:  -1316.401
## Nagelkerke R2:  0.09250297
## % pres/err predicted correctly:  -767.7964
## % of predictable range [ (model-null)/(1-null) ]:  0.05617176
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.9807       0.3169
##
## Degrees of Freedom: 4021 Total (i.e. Null);  4020 Residual
## Null Deviance:       2825
## Residual Deviance: 2676  AIC: 2763
## log likelihood:  -1337.791
## Nagelkerke R2:  0.07230058
## % pres/err predicted correctly:  -779.8097
## % of predictable range [ (model-null)/(1-null) ]:  0.04142342
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.1447      -0.2187
##
## Degrees of Freedom: 4021 Total (i.e. Null);  4020 Residual
```

```
## Null Deviance:       2825
## Residual Deviance: 2813  AIC: 2897
## log likelihood:  -1406.32
## Nagelkerke R2:  0.006110715
## % pres/err predicted correctly:  -810.1125
## % of predictable range [ (model-null)/(1-null) ]:  0.004221546
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.041
##
## Degrees of Freedom: 4021 Total (i.e. Null);  4021 Residual
## Null Deviance:       2825
## Residual Deviance: 2825  AIC: 2911
## log likelihood:  -1412.531
## Nagelkerke R2:  2.200178e-16
## % pres/err predicted correctly:  -813.5512
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.17463     -0.01733
##
## Degrees of Freedom: 4021 Total (i.e. Null);  4020 Residual
## Null Deviance:       2825
## Residual Deviance: 2825  AIC: 2913
## log likelihood:  -1412.353
## Nagelkerke R2:  0.0001748663
## % pres/err predicted correctly:  -813.4505
## % of predictable range [ (model-null)/(1-null) ]:  0.0001236419
## ***************************
```

```
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 ~ CumPres	2676.873	0.00000	1	1	0.1097443	3.245752	30.4867724	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2725.793	48.92031	0	0	0.0925030	0.2102618	NA	NA	-0.0742965	0.8935066	NA
preserved ~ pos	2763.369	86.49563	0	0	0.0723006	0.9806942	NA	NA	NA	0.3168896	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2896.952	220.07862	0	0	0.0061107	1.1446923	NA	-0.2187382	NA	NA	NA
preserved ~ 1	2911.104	234.23132	0	0	0.0000000	0.0411361	NA	NA	NA	NA	NA
preserved ~ stimlen	2912.558	235.68531	0	0	0.0001749	0.1746326	NA	NA	NA	NA	-0.0173252

```
#  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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.2566       0.5536
##
## Degrees of Freedom: 3457 Total (i.e. Null);  3456 Residual
## Null Deviance:       2483
## Residual Deviance: 2287  AIC: 2357
## log likelihood:  -1143.62
## Nagelkerke R2:  0.1073671
## % pres/err predicted correctly:  -675.8694
## % of predictable range [ (model-null)/(1-null) ]:  0.06107383
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)     I(pos^2)          pos
##     0.16785     -0.08054      0.93678
##
## Degrees of Freedom: 3457 Total (i.e. Null);  3455 Residual
## Null Deviance:       2483
## Residual Deviance: 2302  AIC: 2383
## log likelihood:  -1151.081
## Nagelkerke R2:  0.09938949
## % pres/err predicted correctly:  -676.0037
## % of predictable range [ (model-null)/(1-null) ]:  0.06088757
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.9816       0.3084
##
## Degrees of Freedom: 3457 Total (i.e. Null);  3456 Residual
## Null Deviance:       2483
## Residual Deviance: 2349  AIC: 2425
## log likelihood:  -1174.715
## Nagelkerke R2:  0.07389071
## % pres/err predicted correctly:  -688.77
## % of predictable range [ (model-null)/(1-null) ]:  0.04317861
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0566      -0.1334
##
## Degrees of Freedom: 3457 Total (i.e. Null);  3456 Residual
## Null Deviance:       2483
## Residual Deviance: 2480  AIC: 2553
## log likelihood:  -1239.901
## Nagelkerke R2:  0.001728661
## % pres/err predicted correctly:  -718.9642
## % of predictable range [ (model-null)/(1-null) ]:  0.001294501
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.003
##
```

```
## Degrees of Freedom: 3457 Total (i.e. Null);  3457 Residual
## Null Deviance:       2483
## Residual Deviance: 2483  AIC: 2556
## log likelihood:  -1241.433
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -719.8974
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.13969     -0.01777
##
## Degrees of Freedom: 3457 Total (i.e. Null);  3456 Residual
## Null Deviance:       2483
## Residual Deviance: 2483  AIC: 2557
## log likelihood:  -1241.266
## Nagelkerke R2:  0.0001885491
## % pres/err predicted correctly:  -719.7963
## % of predictable range [ (model-null)/(1-null) ]:  0.0001402194
## ***************************
```

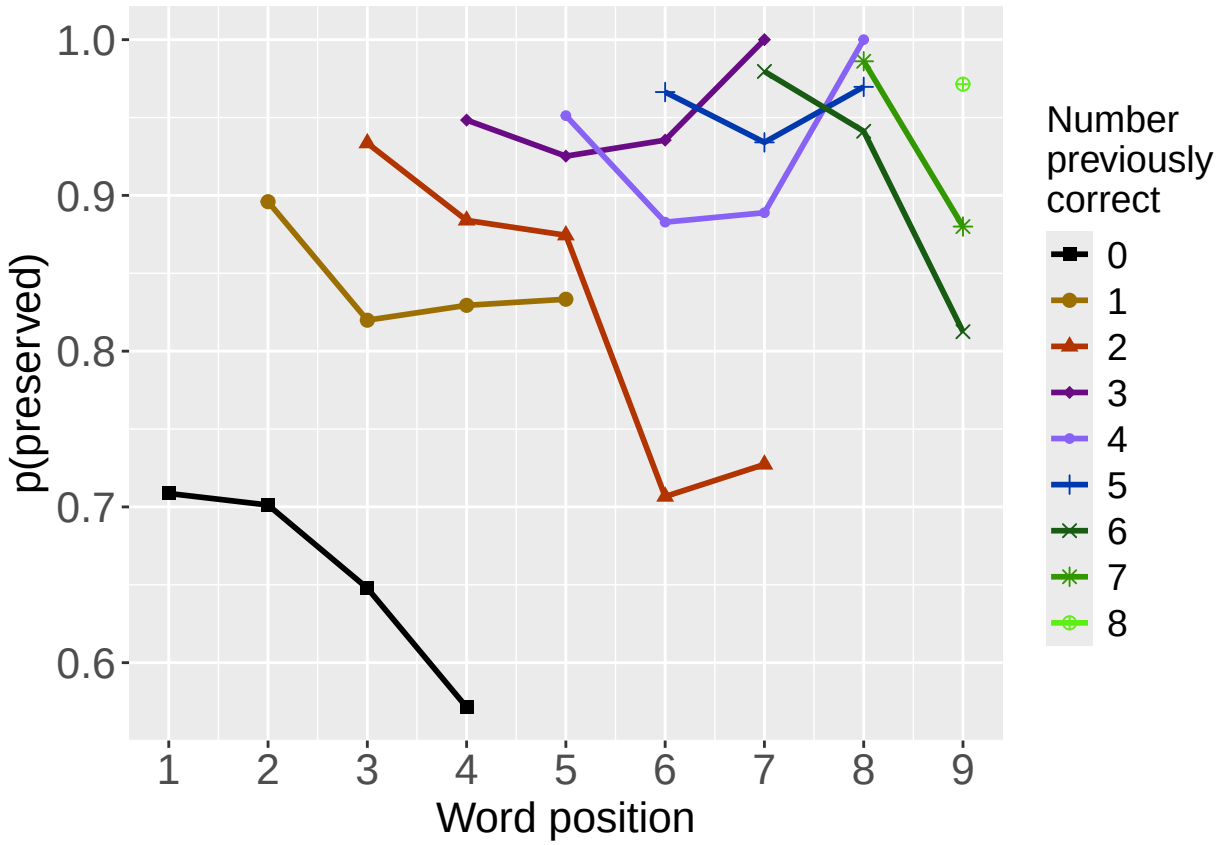
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	2356.598	0.00000	1.0e+00	0.9999977	0.1073671	1.2566096	0.5535705	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2382.571	25.97349	2.3e-06	0.0000023	0.0993895	0.1678524	NA	NA	-0.0805372	0.9367817	NA
preserved ~ pos	2424.581	67.98357	0.0e+00	0.0000000	0.0738907	0.9816499	NA	NA	NA	0.3083826	NA
preserved ~ CumErr	2553.473	196.87499	4.0e+00	0.0000000	0.0017287	0.0566174	NA	-0.1334151	NA	NA	NA
preserved ~ 1	2555.880	199.28186	0.0e+00	0.0000000	0.0000000	0.0030895	NA	NA	NA	NA	NA
preserved ~ stimlen	2557.346	200.74786	0.0e+00	0.0000000	0.0001885	0.1396888	NA	NA	NA	NA	-0.0177655

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

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

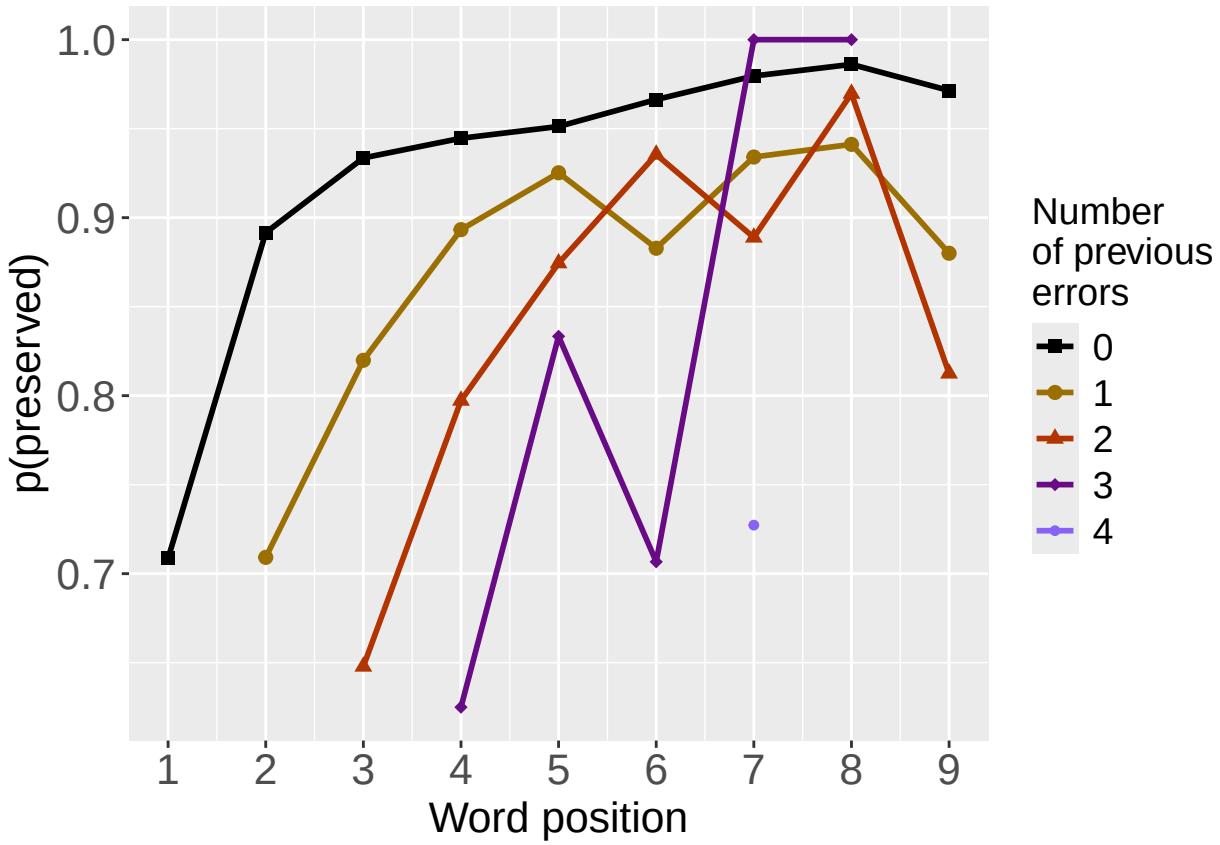
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

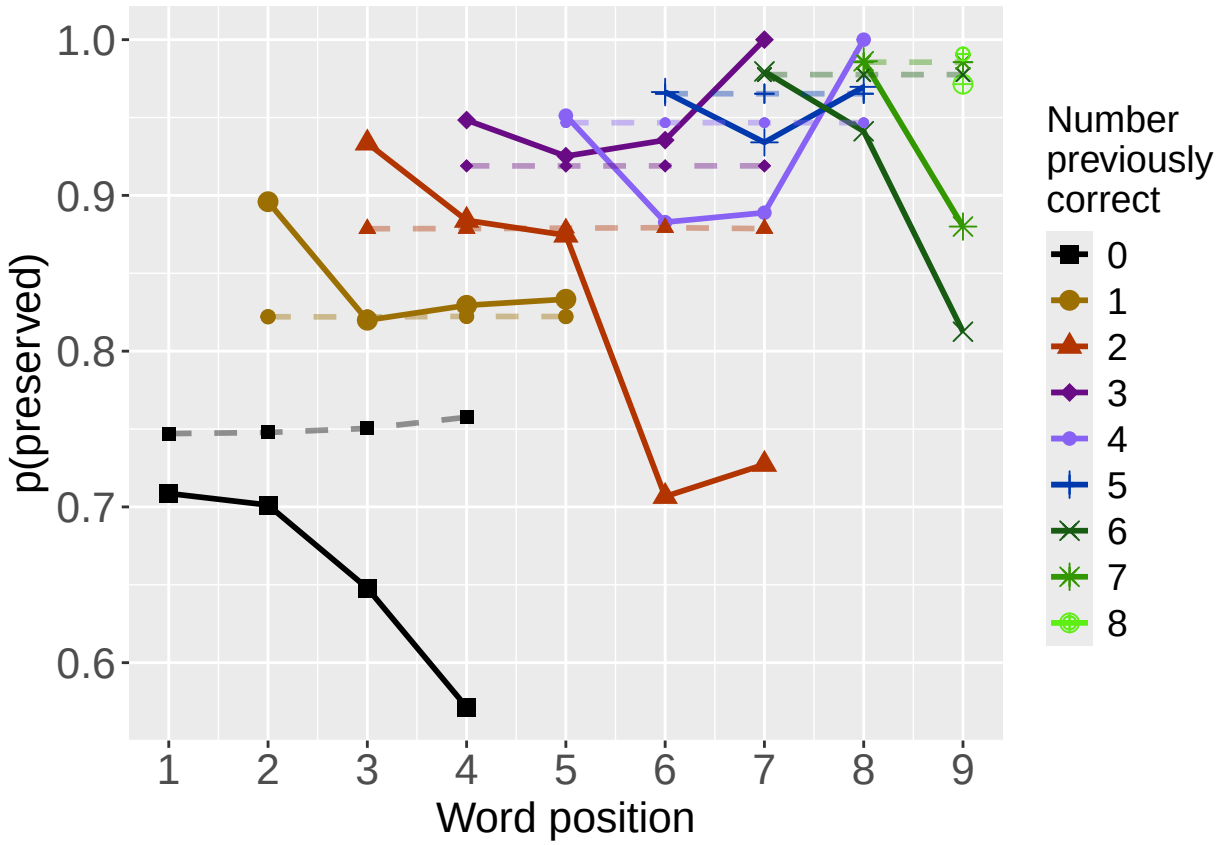
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

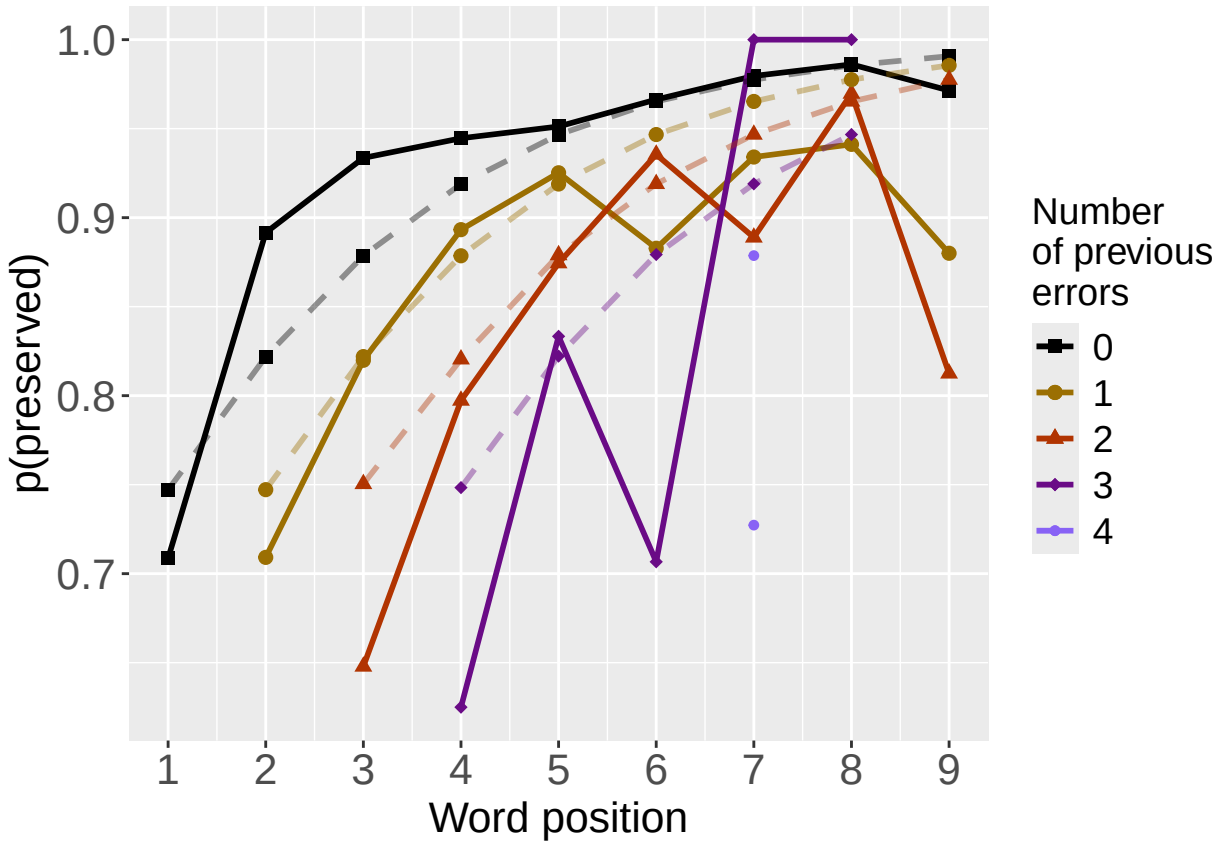
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos
##     0.73247      0.69112     -0.06656      0.28439
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3025  AIC: 3108
## log likelihood:  -1512.695
## Nagelkerke R2:  0.1317332
## % pres/err predicted correctly:  -914.1184
## % of predictable range [ (model-null)/(1-null) ]:  0.07883379
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.0830       0.4485
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3081  AIC: 3161
## log likelihood:  -1540.534
## Nagelkerke R2:  0.1093464
## % pres/err predicted correctly:  -928.473
## % of predictable range [ (model-null)/(1-null) ]:  0.06438434
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.26061     -0.05683      0.74010
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3150  AIC: 3242
## log likelihood:  -1575.139
## Nagelkerke R2:  0.08111066
## % pres/err predicted correctly:  -943.0805
## % of predictable range [ (model-null)/(1-null) ]:  0.04968032
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos
preserved ~ CumPres + I(pos^2) + pos	3107.638	0.0000	1	1	0.1317332	0.7324706	0.6911225	-0.0665632	0.2843949

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(posˆ2)	pos
preserved ~ CumPres	3160.708	53.0707	0	0	0.1093464	1.0830050	0.4484800	NA	NA
preserved ~ I(posˆ2) + pos	3241.816	134.1787	0	0	0.0811107	0.2606136	NA	-0.0568268	0.7400956

```
########
# 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)      CumPres      stimlen
##      2.2136       0.4788      -0.1547
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3049  AIC: 3128
## log likelihood:  -1524.504
## Nagelkerke R2:  0.1222724
## % pres/err predicted correctly:  -917.6936
## % of predictable range [ (model-null)/(1-null) ]:  0.07523492
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.0830       0.4485
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3081  AIC: 3161
## log likelihood:  -1540.534
## Nagelkerke R2:  0.1093464
## % pres/err predicted correctly:  -928.473
## % of predictable range [ (model-null)/(1-null) ]:  0.06438434
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.11555     -0.03204
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3342  AIC: 3429
## log likelihood:  -1670.829
## Nagelkerke R2:  0.000631358
## % pres/err predicted correctly:  -992.0361
## % of predictable range [ (model-null)/(1-null) ]:  0.0004011771
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	stimlen
preserved ~ CumPres + stimlen	3128.122	0.00000	1e+00	0.9999999	0.1222724	2.213621	0.4787724	-0.1547469
preserved ~ CumPres	3160.708	32.58604	1e-07	0.0000001	0.1093464	1.083005	0.4484800	NA
preserved ~ stimlen	3428.505	300.38276	0e+00	0.0000000	0.0006314	2.115551	NA	-0.0320397

```
########
# 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)
}
```

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"pos"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos
##      1.4129       0.6759      -0.2177
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3063  AIC: 3140
## log likelihood:  -1531.281
## Nagelkerke R2:  0.1168195
## % pres/err predicted correctly:  -923.6434
## % of predictable range [ (model-null)/(1-null) ]:  0.0692458
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.0830       0.4485
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3081  AIC: 3161
## log likelihood:  -1540.534
## Nagelkerke R2:  0.1093464
## % pres/err predicted correctly:  -928.473
## % of predictable range [ (model-null)/(1-null) ]:  0.06438434
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.8628       0.2981
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3180  AIC: 3267
## log likelihood:  -1589.796
## Nagelkerke R2:  0.06901415
## % pres/err predicted correctly:  -952.1153
## % of predictable range [ (model-null)/(1-null) ]:  0.04058576
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	pos
preserved ~ CumPres + pos	3140.288	0.00000	1.00e+00	0.9999632	0.1168195	1.4129131	0.6758798	-0.2176817
preserved ~ CumPres	3160.708	20.42015	3.68e-05	0.0000368	0.1093464	1.0830050	0.4484800	NA
preserved ~ pos	3266.739	126.45104	0.00e+00	0.0000000	0.0690141	0.8627698	NA	0.2980631

```
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)	CumPres	I(pos^2)	pos	stimlen
preserved ~ CumPres + I(pos^2) + pos	3107.638	0.00000	1.00e+00	1.0000000	0.1317332	2.7324706	0.6911225	-0.0665632	0.2843949	NA
preserved ~ CumPres + stimlen	3128.122	0.00000	1.00e+00	0.9999999	0.1222722	2.2136207	0.4787724	NA	NA	-0.1547469
preserved ~ CumPres + pos	3140.288	0.00000	1.00e+00	0.9999632	0.1168195	1.4129131	0.6758798	NA	-0.2176817	NA
preserved ~ CumPres	3160.708	53.07070	0.00e+00	0.0000000	0.1093464	1.0830050	0.4484800	NA	NA	NA
preserved ~ CumPres	3160.708	32.58604	1.00e-07	0.0000001	0.1093464	1.0830050	0.4484800	NA	NA	NA
preserved ~ CumPres	3160.708	20.42015	3.68e-05	0.0000368	0.1093464	1.0830050	0.4484800	NA	NA	NA
preserved ~ I(pos^2) + pos	3241.816	134.17868	0.00e+00	0.0000000	0.0811107	0.7260613	NA	-0.0568268	0.7400956	NA
preserved ~ pos	3266.739	126.45104	0.00e+00	0.0000000	0.0690141	0.8627698	NA	NA	0.2980631	NA
preserved ~ stimlen	3428.505	300.38270	0.00e+00	0.0000000	0.0006312	1.1155508	NA	NA	NA	-0.0320397

```
# 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)      CumPres     I(pos^2)          pos      stimlen     log_freq
##     1.39003      0.66771     -0.06239      0.30180     -0.09502      0.08971
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4301 Residual
## Null Deviance:       3343
## Residual Deviance: 2995  AIC: 3079
## log likelihood:  -1497.526
## Nagelkerke R2:  0.1438108
## % pres/err predicted correctly:  -904.5596
## % of predictable range [ (model-null)/(1-null) ]:  0.08845574
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos     log_freq
##     0.70418      0.68197     -0.06642      0.30070      0.11080
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4302 Residual
## Null Deviance:       3343
## Residual Deviance: 3005  AIC: 3089
## log likelihood:  -1502.604
## Nagelkerke R2:  0.1397771
## % pres/err predicted correctly:  -908.755
## % of predictable range [ (model-null)/(1-null) ]:  0.08423263
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos      stimlen
```

```
##     1.60127      0.66903     -0.06128      0.29085     -0.12164
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4302 Residual
## Null Deviance:       3343
## Residual Deviance: 3007  AIC: 3090
## log likelihood:  -1503.681
## Nagelkerke R2:  0.1389209
## % pres/err predicted correctly:  -907.0612
## % of predictable range [ (model-null)/(1-null) ]:  0.0859376
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos
##     0.73247      0.69112     -0.06656      0.28439
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3025  AIC: 3108
## log likelihood:  -1512.695
## Nagelkerke R2:  0.1317332
## % pres/err predicted correctly:  -914.1184
## % of predictable range [ (model-null)/(1-null) ]:  0.07883379
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.868
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4306 Residual
## Null Deviance:       3343
## Residual Deviance: 3343  AIC: 3428
## log likelihood:  -1671.563
## Nagelkerke R2:  2.056531e-16
## % pres/err predicted correctly:  -992.4346
## % 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)
```

```r
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)	CumPres	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumPres + I(pos^2) + pos + stimlen + log_freq	3079.481	0.000000	1.0000000	0.9848704	0.1438108	3.9002770	7.6677069	-0.0623882	0.3017976	0.0897094	-0.0950184
preserved ~ CumPres + I(pos^2) + pos + log_freq	3088.546	9.064231	0.0107579	0.0105952	0.1397771	1.7041778	8.6819740	-0.0664162	0.3007004	0.1108025	NA
preserved ~ CumPres + I(pos^2) + pos + stimlen	3090.245	10.763675	0.0045994	0.0045298	0.1389219	6.6012655	5.6690290	-0.0612792	0.2908465	NA	-0.1216442
preserved ~ CumPres + I(pos^2) + pos	3107.638	28.156218	0.0000008	0.0000008	0.1317332	3.7324706	6.6911225	-0.0665632	0.2843949	NA	NA
preserved ~ 1	3428.347	348.865936	0.0000000	0.0000000	0.0000000	0.8683239	NA	NA	NA	NA	NA

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

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

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

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

```
## Single term deletions
##
## Model:
## preserved ~ CumPres + I(pos^2) + pos + stimlen + log_freq
##           Df Deviance    AIC
## CumPres    1   3110.1 3192.5
## I(pos^2)   1   3027.3 3109.8
## log_freq   1   3007.4 3089.8
## pos        1   3005.3 3087.8
## stimlen    1   3005.2 3087.6
## <none>         2995.1 3079.5
```

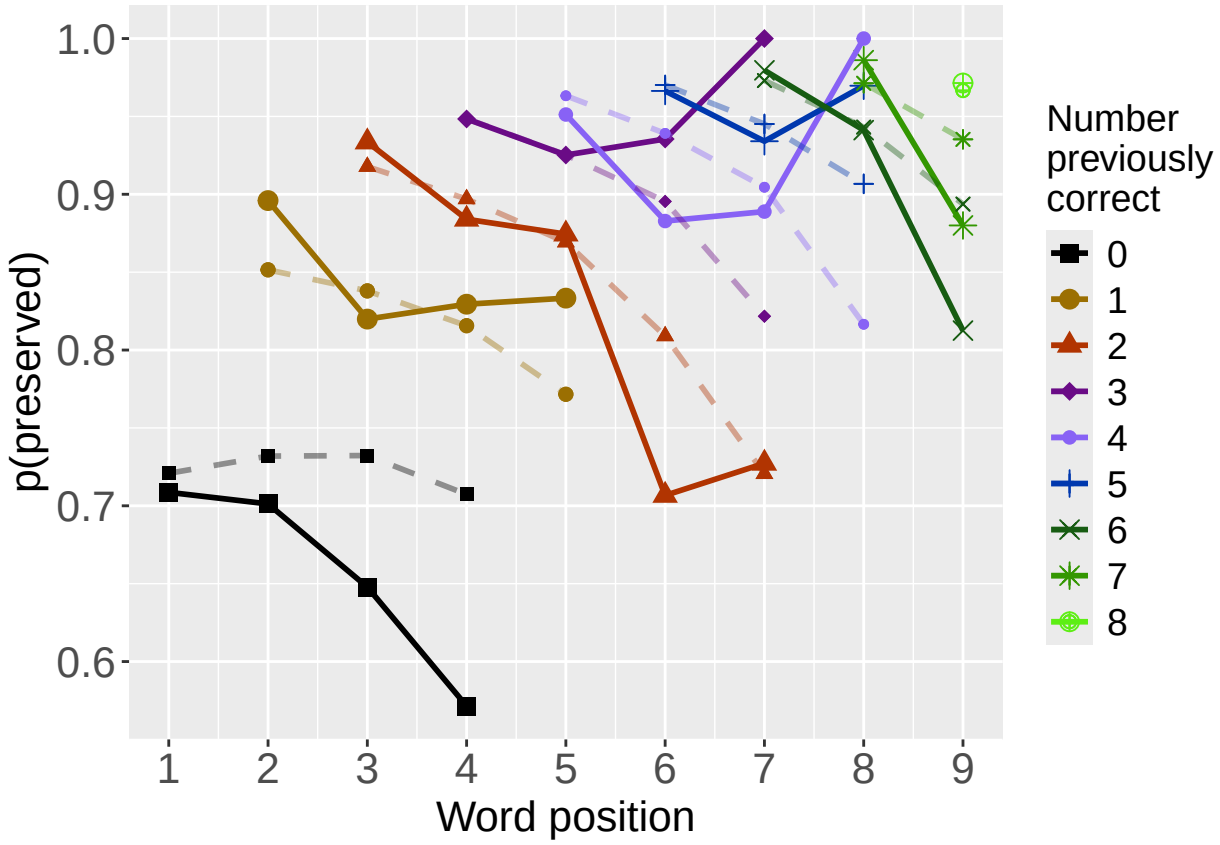
```r
##################################
# Single deletions from best model
```

```
################################

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

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

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

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

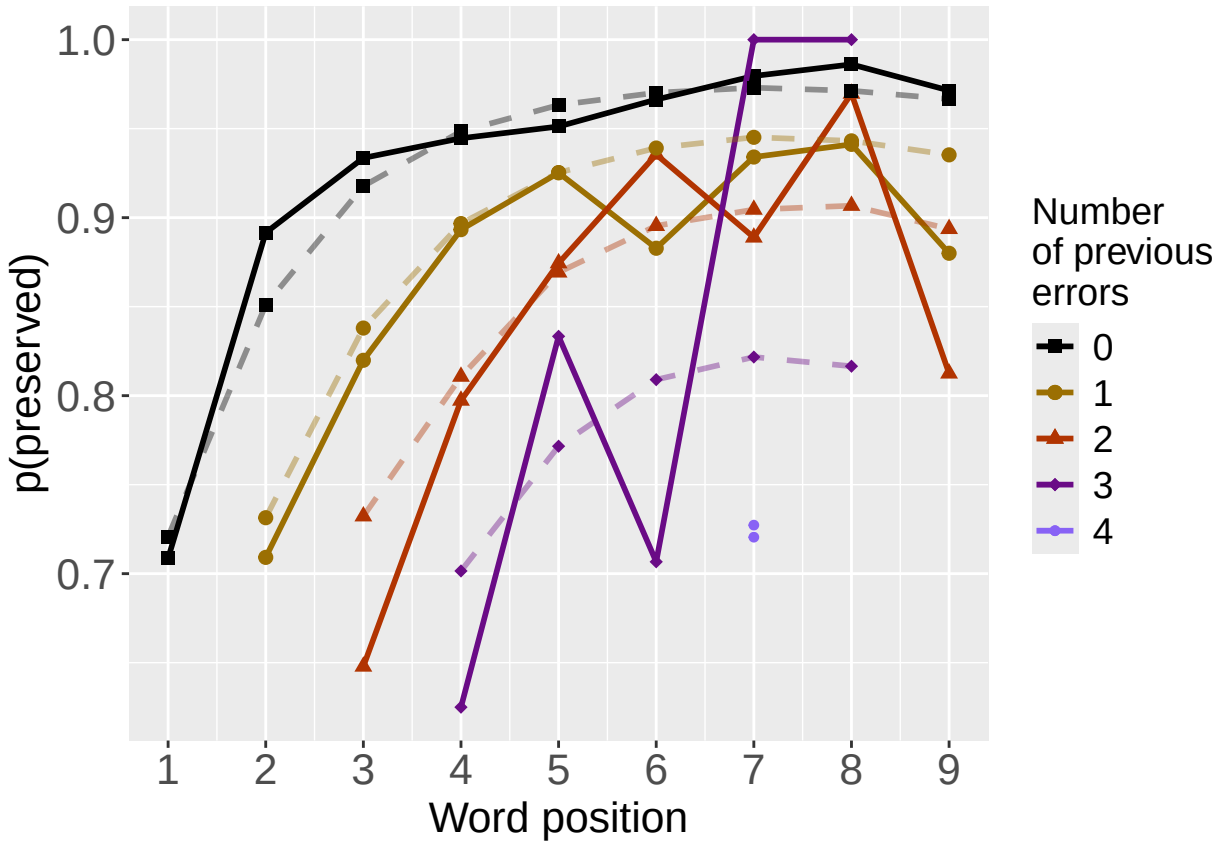
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

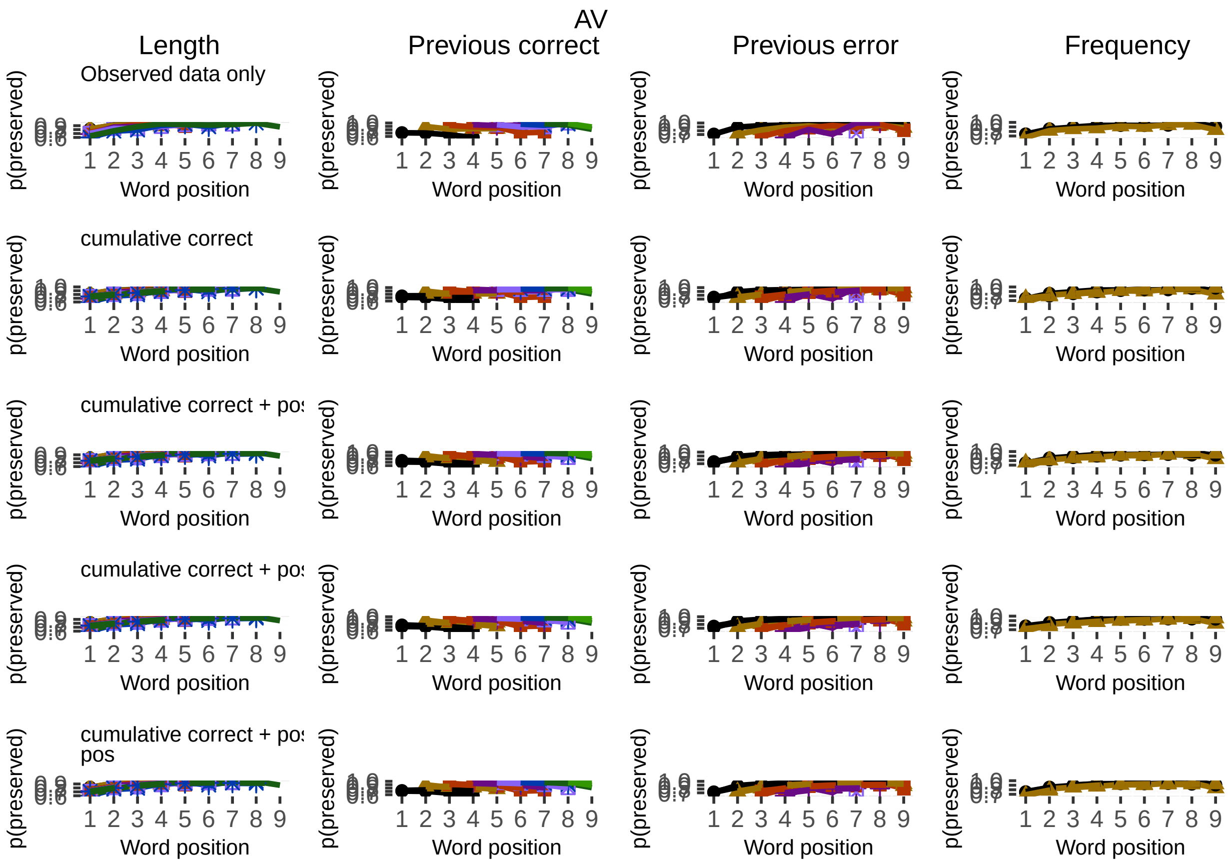
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.0830       0.4485
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4305 Residual
## Null Deviance:       3343
## Residual Deviance: 3081  AIC: 3161
## log likelihood:  -1540.534
```

```
## Nagelkerke R2:  0.1093464
## % pres/err predicted correctly:  -928.473
## % of predictable range [ (model-null)/(1-null) ]:  0.06438434
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)
##     1.13281      0.76026     -0.03879
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4304 Residual
## Null Deviance:       3343
## Residual Deviance: 3035  AIC: 3112
## log likelihood:  -1517.273
## Nagelkerke R2:  0.1280719
## % pres/err predicted correctly:  -916.7218
## % of predictable range [ (model-null)/(1-null) ]:  0.07621319
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)     log_freq
##     1.12801      0.75452     -0.03706      0.10774
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4303 Residual
## Null Deviance:       3343
## Residual Deviance: 3015  AIC: 3094
## log likelihood:  -1507.698
## Nagelkerke R2:  0.1357216
## % pres/err predicted correctly:  -911.8584
## % of predictable range [ (model-null)/(1-null) ]:  0.0811087
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)     log_freq          pos
##     0.70418      0.68197     -0.06642      0.11080      0.30070
##
## Degrees of Freedom: 4306 Total (i.e. Null);  4302 Residual
## Null Deviance:       3343
## Residual Deviance: 3005  AIC: 3089
## log likelihood:  -1502.604
## Nagelkerke R2:  0.1397771
## % pres/err predicted correctly:  -908.755
## % of predictable range [ (model-null)/(1-null) ]:  0.08423263
```

```
## ***************************
## `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()`).
```

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




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

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

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

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

	CumPres	I(pos^2)	pos	stimlen	log_freq
McFadden	0.0547578	0.0188475	0.0217779	0.0045061	0.0048479
SquaredCorrelation	0.0415485	0.0144980	0.0168453	0.0033803	0.0036727
Nagelkerke	0.0415485	0.0144980	0.0168453	0.0033803	0.0036727
Estrella	0.0441179	0.0151307	0.0174548	0.0036419	0.0039082

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

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

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

```
##                                                                 model deviance
## CumPres + I(pos^2) + log_freq + pos CumPres + I(pos^2) + log_freq + pos 3005.208
## CumPres + I(pos^2) + log_freq             CumPres + I(pos^2) + log_freq 3015.395
## CumPres + I(pos^2)                                 CumPres + I(pos^2) 3034.546
## CumPres                                                       CumPres 3081.067
## null                                                             null 3343.126
##                                     deviance_explained percent_explained
## CumPres + I(pos^2) + log_freq + pos           337.9180         10.107845
## CumPres + I(pos^2) + log_freq                 327.7309          9.803127
## CumPres + I(pos^2)                            308.5805          9.230297
## CumPres                                       262.0593          7.838748
## null                                            0.0000          0.000000
##                                     percent_of_explained_deviance increment_in_explained
## CumPres + I(pos^2) + log_freq + pos                     100.00000               3.014666
## CumPres + I(pos^2) + log_freq                            96.98533               5.667186
## CumPres + I(pos^2)                                       91.31815              13.767019
## CumPres                                                  77.55113              77.551129
## null                                                           NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumPres + I(pos^2) + log_freq + pos	3005.208	337.9180
CumPres + I(pos^2) + log_freq	3015.395	327.7309
CumPres + I(pos^2)	3034.546	308.5805
CumPres	3081.067	262.0593
null	3343.126	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumPres + I(pos^2) + log_freq + pos	10.107845	100.00000	3.014666
CumPres + I(pos^2) + log_freq	9.803128	96.98533	5.667186
CumPres + I(pos^2)	9.230297	91.31815	13.767019
CumPres	7.838748	77.55113	77.551129
null	0.000000	NA	0.000000

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

```
##          Nagelkerke
## CumPres  0.51971557
## I(pos^2) 0.18134984
## pos      0.21071170
## stimlen  0.04228245
## log_freq 0.04594044
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumPres	0.7442191	3081.067
preserved ~ CumPres+I(pos^2)	0.7963693	3034.546
preserved ~ CumPres+I(pos^2)+log_freq	0.8114531	3015.395
preserved ~ CumPres+I(pos^2)+log_freq+pos	0.8491341	3005.208

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

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

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

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

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

```
##                                     model p_accounted_for model_deviance diff_CumPres
## 1                        preserved ~ CumPres       0.7442191       3081.067   0.00000000
## 2              preserved ~ CumPres+I(pos^2)       0.7963693       3034.546   0.05215021
## 3     preserved ~ CumPres+I(pos^2)+log_freq       0.8114531       3015.395   0.06723397
## 4 preserved ~ CumPres+I(pos^2)+log_freq+pos       0.8491341       3005.208   0.10491502
##   diff_CumPres+I(pos^2) diff_CumPres+I(pos^2)+log_freq diff_CumPres+I(pos^2)+log_freq+pos
## 1           -0.05215021                     -0.06723397                       -0.10491502
## 2            0.00000000                     -0.01508376                       -0.05276481
## 3            0.01508376                      0.00000000                       -0.03768106
## 4            0.05276481                      0.03768106                        0.00000000
```

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

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

model	diff_CumPres	diff_CumPres+I(pos^2)	diff_CumPres+I(pos^2)+log_freq
preserved ~ CumPres	0.0000000	-0.0521502	-0.0672340
preserved ~ CumPres+I(pos^2)	0.0521502	0.0000000	-0.0150838
preserved ~ CumPres+I(pos^2)+log_freq	0.0672340	0.0150838	0.0000000
preserved ~ CumPres+I(pos^2)+log_freq+pos	0.1049150	0.0527648	0.0376811

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