AC - 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	524	29	126	NA	NA	679
2	60	NA	420	95	104	679
3	301	NA	160	203	15	679
4	291	NA	229	64	37	621
5	225	NA	204	65	36	530
6	196	1	131	67	22	417
7	166	NA	96	29	18	309
8	85	NA	50	25	3	163
9	71	NA	2	NA	7	80

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7717231	0.0427099	0.1855670	NA	NA	679
2	0.0883652	NA	0.6185567	0.1399116	0.1531664	679
3	0.4432990	NA	0.2356406	0.2989691	0.0220913	679
4	0.4685990	NA	0.3687601	0.1030596	0.0595813	621
5	0.4245283	NA	0.3849057	0.1226415	0.0679245	530
6	0.4700240	0.0023981	0.3141487	0.1606715	0.0527578	417

pos_factor	O	P	V	1	S	total
7	0.5372168	NA	0.3106796	0.0938511	0.0582524	309
8	0.5214724	NA	0.3067485	0.1533742	0.0184049	163
9	0.8875000	NA	0.0250000	NA	0.0875000	80

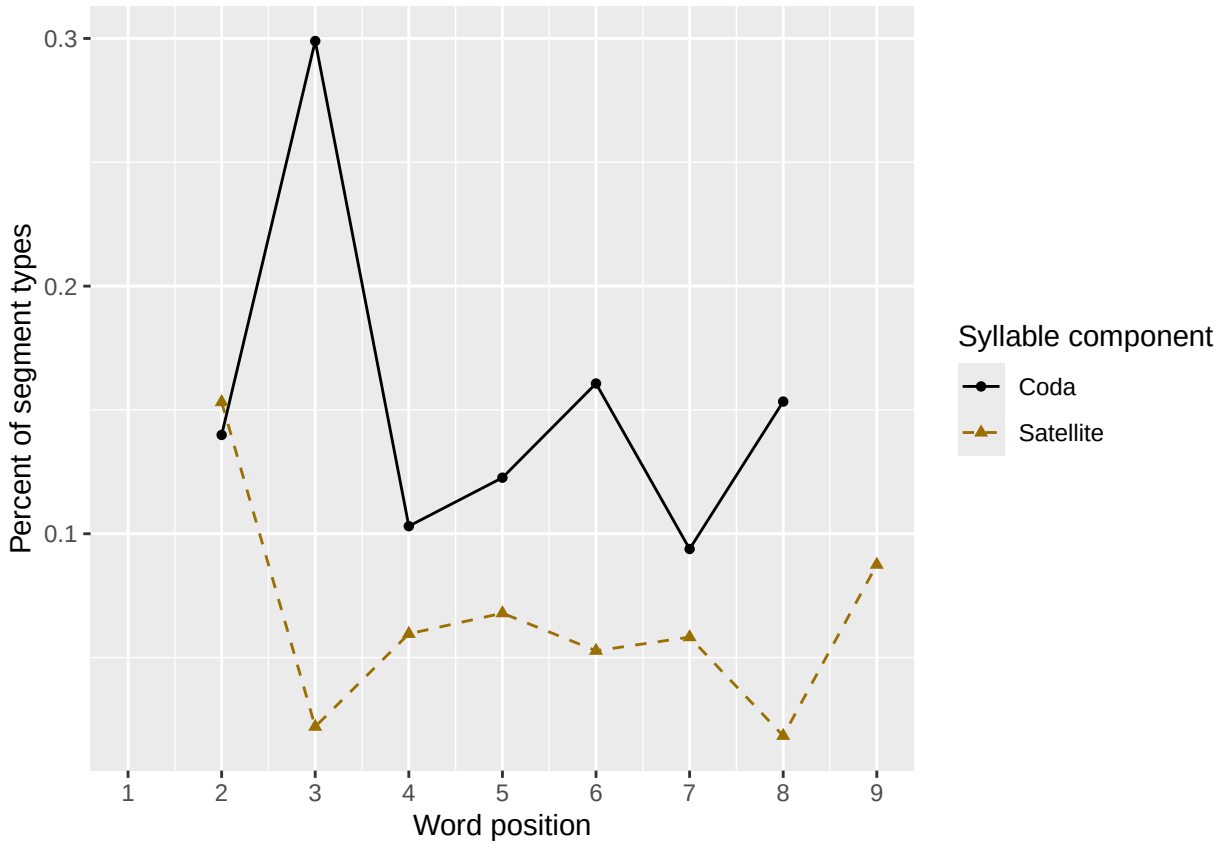
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 1     1     1     NA     NA     NA     NA     NA     NA
## 2       5 0.912 0.967 0.978  0.989 NA     NA     NA     NA     NA
## 3       6 0.947 0.965 0.956  0.956  0.956 NA     NA     NA     NA
## 4       7 0.935 0.968 0.968  0.981  0.986  0.963 NA     NA     NA
## 5       8 0.932 0.969 0.961  0.928  0.945  0.945  0.943 NA     NA
## 6       9 0.873 0.946 0.958  0.934  0.964  0.952  0.970  0.982 NA
## 7      10 0.924 0.941 0.912  0.934  0.908  0.888  0.888  0.929  0.944
```

```
# 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.
```

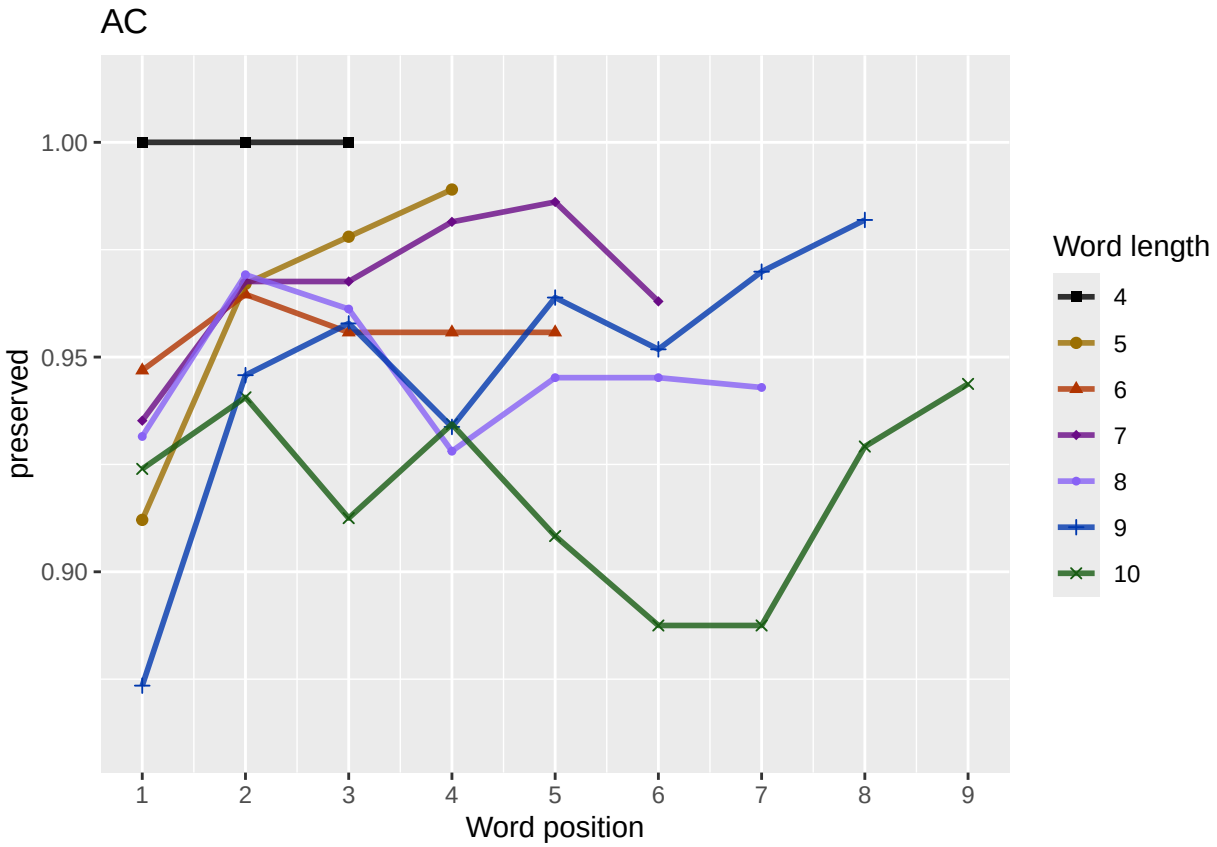
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    58    58    58    NA    NA    NA    NA    NA    NA
## 2       5    91    91    91    91    NA    NA    NA    NA    NA
## 3       6   113   113   113   113   113    NA    NA    NA    NA
## 4       7   108   108   108   108   108   108    NA    NA    NA
## 5       8   146   146   146   146   146   146   146    NA    NA
## 6       9    83    83    83    83    83    83    83    83    NA
## 7      10    80    80    80    80    80    80    80    80    80
```

```
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:  4
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.59503     -0.23856      0.05985
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1580  AIC: 1710
## log likelihood:  -789.8145
## Nagelkerke R2:  0.01949497
## % pres/err predicted correctly:  -391.3547
## % of predictable range [ (model-null)/(1-null) ]:  0.007278528
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.73627     -0.13837      0.34725     -0.03263
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1578  AIC: 1710
## log likelihood:  -788.8824
## Nagelkerke R2:  0.02088564
## % pres/err predicted correctly:  -391.1934
## % of predictable range [ (model-null)/(1-null) ]:  0.007686558
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.6014      -0.2103
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1583  AIC: 1711
## log likelihood:  -791.3604
## Nagelkerke R2:  0.01718728
## % pres/err predicted correctly:  -391.7077
## % of predictable range [ (model-null)/(1-null) ]:  0.006385407
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.36751     -0.23190     -0.01299      0.17163
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1579  AIC: 1712
## log likelihood:  -789.4013
## Nagelkerke R2:  0.02011155
## % pres/err predicted correctly:  -391.3092
## % of predictable range [ (model-null)/(1-null) ]:  0.00739365
## ***************************
## 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)
##         2.388794          0.009572         -0.152019          1.403522          0.016253
##      stimlen:pos
##        -0.146847
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4151 Residual
## Null Deviance:       1606
## Residual Deviance: 1576  AIC: 1713
## log likelihood:  -787.8557
## Nagelkerke R2:  0.02241654
## % pres/err predicted correctly:  -391.0992
## % of predictable range [ (model-null)/(1-null) ]:  0.007924923
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.938
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4156 Residual
## Null Deviance:       1606
## Residual Deviance: 1606  AIC: 1736
## log likelihood:  -802.8382
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -394.2314
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.61882     -0.02443      0.20780
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1603  AIC: 1737
## log likelihood:  -801.3406
## Nagelkerke R2:  0.00224784
## % pres/err predicted correctly:  -393.9649
## % of predictable range [ (model-null)/(1-null) ]:  0.0006743267
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##    2.956584    -0.004792
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1606  AIC: 1738
## log likelihood:  -802.8277
## Nagelkerke R2:  1.566305e-05
## % pres/err predicted correctly:  -394.2285
## % of predictable range [ (model-null)/(1-null) ]:  7.345e-06
## ***************************
```

```
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 + pos	1709.92	0.000000	1.000000	0.3303789	0.0194957	0.595035	- 0.2385586	0.0598499	NA	NA	NA
preserved ~ stimlen * pos	1710.34	0.420980	0.810187	0.2676687	0.0208836	0.736270	- 0.1383669	0.3472493	- 0.0326327	NA	NA
preserved ~ stimlen	1711.08	1.165959	0.558232	0.1844283	0.0171874	0.601445	- 0.2102637	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	1711.58	4.662152	0.4355804	0.1439066	0.6020114	4.6367507	-0.2318963	0.1716277	NA	-0.0129852	NA
preserved ~ stimlen * (I(pos^2) + pos)	1712.92	6.002757	0.2228228	0.0736159	0.6224163	4.5388794	0.0095724	0.4403521	-0.1468467	-0.1520194	0.016253
preserved ~ 1	1735.61	28.692858	0.0000000	0.0000000	0.0000000	2.0938210	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1737.30	30.382552	0.0000000	0.0000000	0.0022447	2.8618822	NA	0.2077965	NA	-0.0244347	NA
preserved ~ pos	1737.56	30.637866	0.0000000	0.0000000	0.0000127	2.7956584	NA	-0.0047924	NA	NA	NA

```
print(BestLPModelFormula)
```

```
## [1] "preserved ~ stimlen + pos"
```

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.59503     -0.23856      0.05985
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1580  AIC: 1710
```

```
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
```

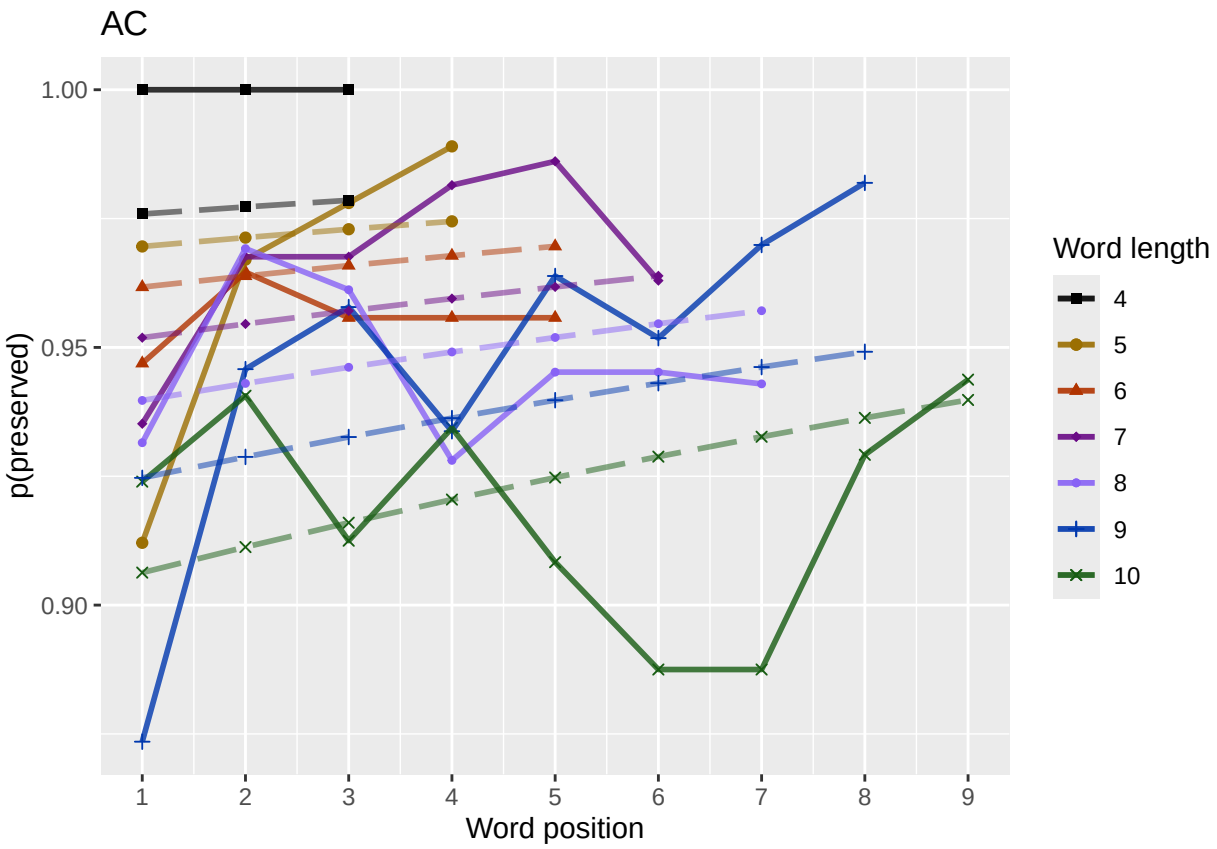
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.976 0.977 0.979 NA     NA     NA     NA     NA     NA
## 2       5 0.970 0.971 0.973  0.974 NA     NA     NA     NA     NA
## 3       6 0.962 0.964 0.966  0.968  0.970 NA     NA     NA     NA
## 4       7 0.952 0.955 0.957  0.959  0.962  0.964 NA     NA     NA
## 5       8 0.940 0.943 0.946  0.949  0.952  0.955  0.957 NA     NA
## 6       9 0.925 0.929 0.933  0.936  0.940  0.943  0.946  0.949 NA
```

```
## 7      10 0.906 0.911 0.916  0.920  0.925  0.929  0.933  0.936  0.940
```

```
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        6   679
```

```
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 6 / 679 = 0.88 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:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
```

```
##      4.6534      -0.2539       0.1012
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4136 Residual
## Null Deviance:       1499
## Residual Deviance: 1470  AIC: 1601
## log likelihood:  -735.2246
## Nagelkerke R2:  0.02275212
## % pres/err predicted correctly:  -359.3376
## % of predictable range [ (model-null)/(1-null) ]:  0.008021324
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          1.52864           0.10045          -0.27446           2.26194           0.02987
##      stimlen:pos
##         -0.23933
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4133 Residual
## Null Deviance:       1499
## Residual Deviance: 1463  AIC: 1601
## log likelihood:  -731.6611
## Nagelkerke R2:  0.02837484
## % pres/err predicted correctly:  -358.9608
## % of predictable range [ (model-null)/(1-null) ]:  0.009058476
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.91026     -0.16716      0.35826     -0.02924
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4135 Residual
## Null Deviance:       1499
## Residual Deviance: 1469  AIC: 1602
## log likelihood:  -734.5692
## Nagelkerke R2:  0.02378695
## % pres/err predicted correctly:  -359.2803
## % of predictable range [ (model-null)/(1-null) ]:  0.008179083
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.47201     -0.24873     -0.01095      0.19367
```

```
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4135 Residual
## Null Deviance:       1499
## Residual Deviance: 1470  AIC: 1603
## log likelihood:  -734.9678
## Nagelkerke R2:  0.02315764
## % pres/err predicted correctly:  -359.3144
## % of predictable range [ (model-null)/(1-null) ]:  0.00808519
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.6744      -0.2084
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4137 Residual
## Null Deviance:       1499
## Residual Deviance: 1478  AIC: 1607
## log likelihood:  -739.2415
## Nagelkerke R2:  0.01640221
## % pres/err predicted correctly:  -360.1814
## % of predictable range [ (model-null)/(1-null) ]:  0.005698291
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.027
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4138 Residual
## Null Deviance:       1499
## Residual Deviance: 1499  AIC: 1629
## log likelihood:  -749.5817
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -362.2514
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.89329      0.03569
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4137 Residual
## Null Deviance:       1499
```

```
## Residual Deviance: 1498  AIC: 1630
## log likelihood:  -749.0645
## Nagelkerke R2:  0.0008222631
## % pres/err predicted correctly:  -362.1621
## % of predictable range [ (model-null)/(1-null) ]:  0.0002456334
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.58929     -0.02316      0.23280
##
## Degrees of Freedom: 4138 Total (i.e. Null);  4136 Residual
## Null Deviance:       1499
## Residual Deviance: 1496  AIC: 1630
## log likelihood:  -747.902
## Nagelkerke R2:  0.002670039
## % pres/err predicted correctly:  -361.9554
## % of predictable range [ (model-null)/(1-null) ]:  0.0008148041
## ***************************
```

```
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 + pos	1600.870	0.0000000	1.0000000	0.3396159	0.0227541	1.1653440	-0.2538551	0.1011944	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1601.180	0.3101564	0.8563482	0.2908295	0.0283748	1.5286420	0.1004482	0.2619394	-0.2393317	-0.2744566	0.0298731
preserved ~ stimlen * pos	1601.776	0.9057022	0.6358128	0.2159322	0.0237873	0.9910256	-0.1671571	0.3582649	-0.0292385	NA	NA
preserved ~ stimlen + I(pos^2) + pos	1602.707	1.8373412	0.3990492	0.1355285	0.0231574	1.6472008	-0.2487309	0.1936744	NA	-0.0109505	NA
preserved ~ stimlen	1606.734	5.8639845	0.0532908	0.0180984	0.0164042	0.2674409	-0.2083700	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(posˆ2)	stimlen:I(posˆ2)
preserved ~ 1	1628.752	27.88377	7.000000e-07	9.000000e-07	0.3000000	3.0026863	NA	NA	NA	NA	NA
preserved ~ pos	1630.052	29.18325	2.000000e-07	5.000000e-07	0.2000822	2.3893293	NA	0.0356880	NA	NA	NA
preserved ~ I(posˆ2) + pos	1630.302	29.43273	1.000000e-07	4.000000e-07	0.1002670	2.0589288	NA	0.2328010	NA	-0.0231619	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.
```

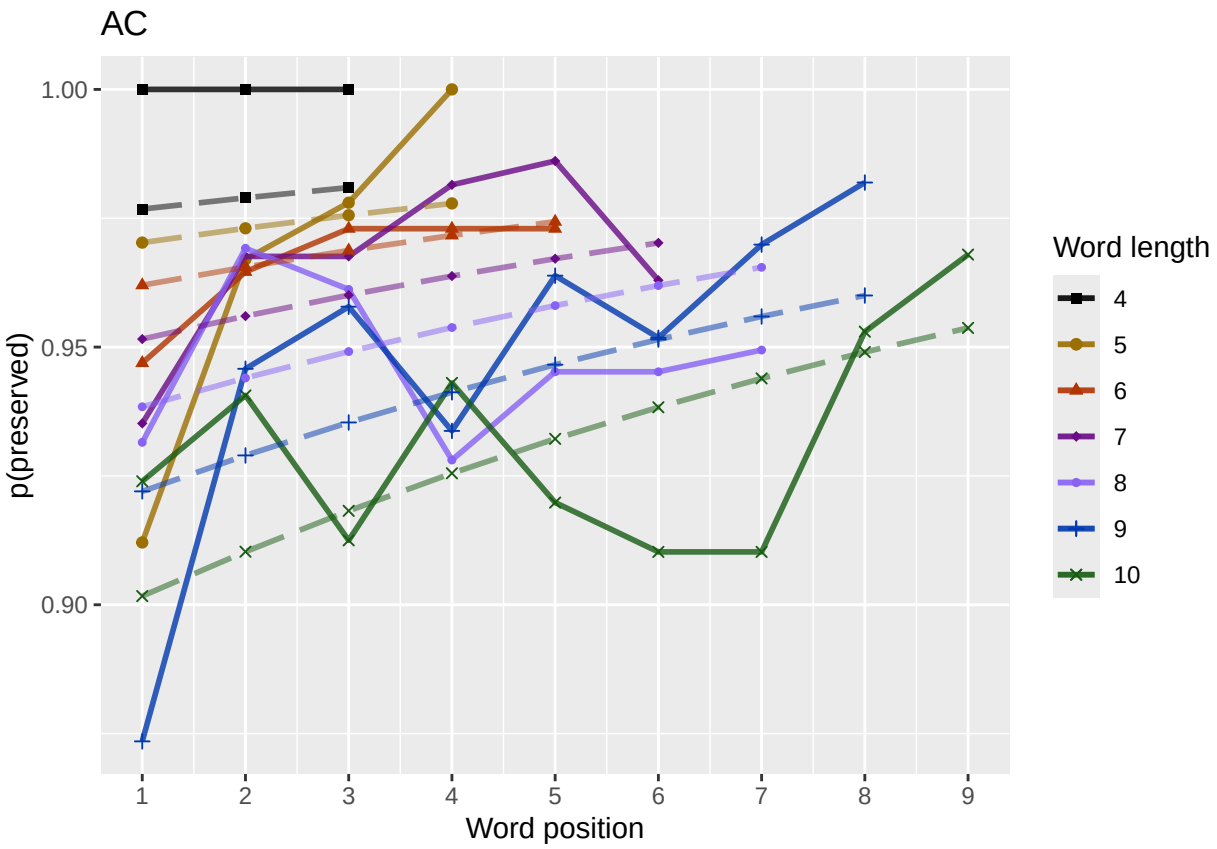
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.977 0.979 0.981 NA     NA     NA     NA     NA     NA
## 2       5 0.970 0.973 0.976  0.978 NA     NA     NA     NA     NA
## 3       6 0.962 0.966 0.969  0.972  0.974 NA     NA     NA     NA
## 4       7 0.952 0.956 0.960  0.964  0.967  0.970 NA     NA     NA
## 5       8 0.938 0.944 0.949  0.954  0.958  0.962  0.965 NA     NA
## 6       9 0.922 0.929 0.935  0.941  0.947  0.951  0.956  0.960 NA
## 7      10 0.902 0.910 0.918  0.926  0.932  0.938  0.944  0.949  0.954
```

```
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: 0.86 - 1.01"
```

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

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

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

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

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

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.0101856
```

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

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

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

```r
print(OA_mean_pos_diff)
```

```
## [1] 0.003048811
```

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

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

```
    potential_u_shape <- FALSE
  }else{
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 1)

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

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

```
## [1] "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:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen          log_freq              pos  stimlen:log_freq
##         4.51521         -0.22331           0.86766          0.06021          -0.08437
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4152 Residual
## Null Deviance:       1606
## Residual Deviance: 1548  AIC: 1680
## log likelihood:  -773.7877
## Nagelkerke R2:  0.04331839
## % pres/err predicted correctly:  -388.3311
## % of predictable range [ (model-null)/(1-null) ]:  0.01492859
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          4.11337          -0.21765           0.61624          -0.02381           0.27902
##  stimlen:log_freq  log_freq:I(pos^2)      log_freq:pos
##         -0.08703          -0.01457           0.14690
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4149 Residual
## Null Deviance:       1606
## Residual Deviance: 1542  AIC: 1680
```

```
## log likelihood:  -770.7619
## Nagelkerke R2:  0.04779549
## % pres/err predicted correctly:  -387.8112
## % of predictable range [ (model-null)/(1-null) ]:  0.01624403
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          4.50858          -0.22938           0.86291           0.07572          -0.09393
##     log_freq:pos
##          0.02157
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4151 Residual
## Null Deviance:       1606
## Residual Deviance: 1546  AIC: 1680
## log likelihood:  -773.1021
## Nagelkerke R2:  0.04433338
## % pres/err predicted correctly:  -388.1891
## % of predictable range [ (model-null)/(1-null) ]:  0.01528794
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          4.52201          -0.19489           0.86743          -0.08436
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1551  AIC: 1681
## log likelihood:  -775.3429
## Nagelkerke R2:  0.04101457
## % pres/err predicted correctly:  -388.6739
## % of predictable range [ (model-null)/(1-null) ]:  0.01406137
## **************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          4.24736          -0.21508           0.86918          -0.01514           0.19023
## stimlen:log_freq
##         -0.08449
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4151 Residual
## Null Deviance:       1606
```

```
## Residual Deviance: 1546  AIC: 1681
## log likelihood:  -773.2359
## Nagelkerke R2:  0.04413534
## % pres/err predicted correctly:  -388.2959
## % of predictable range [ (model-null)/(1-null) ]:  0.01501777
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           3.86913           -0.17969           -0.02534            0.27627           -0.06085
## I(pos^2):log_freq       pos:log_freq
##          -0.01764            0.15346
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4150 Residual
## Null Deviance:       1606
## Residual Deviance: 1552  AIC: 1687
## log likelihood:  -776.0956
## Nagelkerke R2:  0.03989907
## % pres/err predicted correctly:  -388.8305
## % of predictable range [ (model-null)/(1-null) ]:  0.01366512
## ****************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     4.26838     -0.18968      0.06027      0.17571
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1559  AIC: 1687
## log likelihood:  -779.385
## Nagelkerke R2:  0.03501903
## % pres/err predicted correctly:  -389.4857
## % of predictable range [ (model-null)/(1-null) ]:  0.01200744
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      4.2751      -0.1612       0.1755
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1562  AIC: 1689
```

```
## log likelihood:  -780.9416
## Nagelkerke R2:  0.03270694
## % pres/err predicted correctly:  -389.7737
## % of predictable range [ (model-null)/(1-null) ]:  0.01127871
## **************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##      4.0238      -0.1823      -0.0139       0.1797       0.1762
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4152 Residual
## Null Deviance:       1606
## Residual Deviance: 1558  AIC: 1689
## log likelihood:  -778.9174
## Nagelkerke R2:  0.03571314
## % pres/err predicted correctly:  -389.4787
## % of predictable range [ (model-null)/(1-null) ]:  0.01202509
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##    4.2675347    -0.1899412     0.0611169     0.1718620     0.0009994
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4152 Residual
## Null Deviance:       1606
## Residual Deviance: 1559  AIC: 1689
## log likelihood:  -779.3833
## Nagelkerke R2:  0.03502149
## % pres/err predicted correctly:  -389.4837
## % of predictable range [ (model-null)/(1-null) ]:  0.01201236
## **************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##          2.518883          -0.034487           0.306646          -0.009173          -0.018216
##      pos:log_freq
##          0.151500
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4151 Residual
## Null Deviance:       1606
## Residual Deviance: 1565  AIC: 1700
## log likelihood:  -782.5439
```

```
## Nagelkerke R2:  0.03032536
## % pres/err predicted correctly:  -390.0179
## % of predictable range [ (model-null)/(1-null) ]:  0.01066076
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.95431      0.01487      0.21122
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1574  AIC: 1702
## log likelihood:  -786.9027
## Nagelkerke R2:  0.02383691
## % pres/err predicted correctly:  -390.8073
## % of predictable range [ (model-null)/(1-null) ]:  0.008663404
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     2.969782      0.010246      0.232469     -0.005603
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1574  AIC: 1704
## log likelihood:  -786.8524
## Nagelkerke R2:  0.02391194
## % pres/err predicted correctly:  -390.8001
## % of predictable range [ (model-null)/(1-null) ]:  0.008681772
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.59503     -0.23856      0.05985
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1580  AIC: 1710
## log likelihood:  -789.8145
## Nagelkerke R2:  0.01949497
## % pres/err predicted correctly:  -391.3547
## % of predictable range [ (model-null)/(1-null) ]:  0.007278528
```

```
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.73627     -0.13837      0.34725     -0.03263
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1578  AIC: 1710
## log likelihood:  -788.8824
## Nagelkerke R2:  0.02088564
## % pres/err predicted correctly:  -391.1934
## % of predictable range [ (model-null)/(1-null) ]:  0.007686558
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.6014      -0.2103
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1583  AIC: 1711
## log likelihood:  -791.3604
## Nagelkerke R2:  0.01718728
## % pres/err predicted correctly:  -391.7077
## % of predictable range [ (model-null)/(1-null) ]:  0.006385407
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.36751     -0.23190     -0.01299      0.17163
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1579  AIC: 1712
## log likelihood:  -789.4013
## Nagelkerke R2:  0.02011155
## % pres/err predicted correctly:  -391.3092
## % of predictable range [ (model-null)/(1-null) ]:  0.00739365
## ***************************
## 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)
##         2.388794          0.009572         -0.152019          1.403522          0.016253
##      stimlen:pos
##        -0.146847
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4151 Residual
## Null Deviance:       1606
## Residual Deviance: 1576  AIC: 1713
## log likelihood:  -787.8557
## Nagelkerke R2:  0.02241654
## % pres/err predicted correctly:  -391.0992
## % of predictable range [ (model-null)/(1-null) ]:  0.007924923
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.938
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4156 Residual
## Null Deviance:       1606
## Residual Deviance: 1606  AIC: 1736
## log likelihood:  -802.8382
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -394.2314
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.61882     -0.02443      0.20780
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1603  AIC: 1737
## log likelihood:  -801.3406
## Nagelkerke R2:  0.00224784
## % pres/err predicted correctly:  -393.9649
## % of predictable range [ (model-null)/(1-null) ]:  0.0006743267
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##    2.956584    -0.004792
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1606  AIC: 1738
## log likelihood:  -802.8277
## Nagelkerke R2:  1.566305e-05
## % pres/err predicted correctly:  -394.2285
## % of predictable range [ (model-null)/(1-null) ]:  7.345e-06
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]:I(pos^2)
preserved ~ stimlen * log_freq + pos	1679.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2233056	0.8676590	-0.0843709	0.0602123	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	1679.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2176546	0.6162379	-0.0870328	0.2790239	NA	0.1468959	-0.0238088	NA	-0.014567	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	1680.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2293810	0.8629122	-0.0939308	0.0757231	NA	0.0215721	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	1680.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1948901	0.8674341	-0.0843576	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	[illegible]	I(pos^2)	I(pos^2):log_freq	[illegible]	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + I(pos^2) + pos	1681.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2150849	0.8691762	-0.0844917	0.1902276	NA	NA	-0.0151380	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	1687.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen + pos + log_freq	1687.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1896832	0.1757094	NA	0.0602661	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	1688.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1612301	0.1755346	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	1688.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1823209	0.1761999	NA	0.1797301	NA	NA	-0.0138965	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	1689.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1899412	0.1718620	NA	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	1700.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0091733	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ pos + log_freq	1702.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.2112240	NA	0.0148665	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	1704.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.2324687	NA	0.0102461	-0.0056028	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	1709.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2385586	NA	NA	0.0598499	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	1710.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1383669	NA	NA	0.3472493	NA	NA	NA	NA	NA	-0.0326327	NA
preserved ~ stimlen	1711.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2102637	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	1711.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2318963	NA	NA	0.1716217	NA	NA	-0.0129852	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1712.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	1.4035218	NA	NA	-0.1520194	NA	NA	-0.1468465	0.016253
preserved ~ 1	1735.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAIC expAICw NagR2 (Intercept) stimlen log_freq stimlen:pos log_freq:pos log_freq:I(pos^2) I(pos^2) stimlen:I(pos^2)								
preserved ~ I(pos^2) + pos	1737.3704533706500000000.0022471882	NA	NA	0.2077965 NA	NA	-0.0244347	NA	NA	NA	NA
preserved ~ pos	1737.5767890800000000000.001957584	NA	NA	-0.0047924	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

```
## [1] "preserved ~ stimlen * log_freq + pos"
```

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          4.51521          -0.22331           0.86766           0.06021          -0.08437
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4152 Residual
## Null Deviance:       1606
## Residual Deviance: 1548  AIC: 1680
```

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

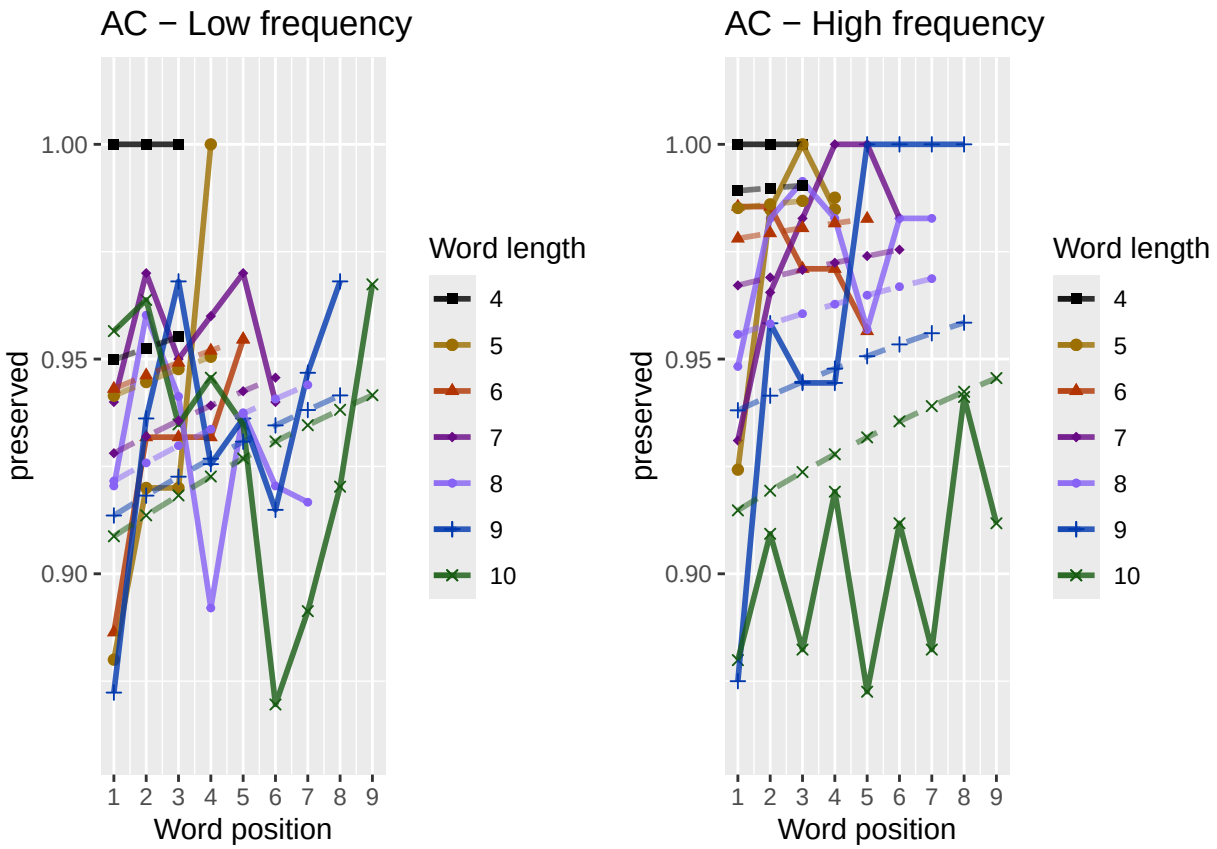
```r
PosDat$FLPFitted<-fitted(BestFLPModel)

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

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

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

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

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

```r
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_frequency_effect_length_pos_wfit.png"),device="png",unit="cm"
print(Both_Plots)
```




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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.338       -1.076
```

```
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1414  AIC: 1527
## log likelihood:  -707.0565
## Nagelkerke R2:  0.1405609
## % pres/err predicted correctly:  -353.2909
## % of predictable range [ (model-null)/(1-null) ]:  0.103586
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.4633       0.2081
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1574  AIC: 1705
## log likelihood:  -787.236
## Nagelkerke R2:  0.02334034
## % pres/err predicted correctly:  -391.0517
## % of predictable range [ (model-null)/(1-null) ]:  0.008045209
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.6014      -0.2103
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1583  AIC: 1711
## log likelihood:  -791.3604
## Nagelkerke R2:  0.01718728
## % pres/err predicted correctly:  -391.7077
## % of predictable range [ (model-null)/(1-null) ]:  0.006385407
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.938
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4156 Residual
## Null Deviance:       1606
```

```
## Residual Deviance: 1606  AIC: 1736
## log likelihood:  -802.8382
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -394.2314
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.61882     -0.02443      0.20780
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1603  AIC: 1737
## log likelihood:  -801.3406
## Nagelkerke R2:  0.00224784
## % pres/err predicted correctly:  -393.9649
## % of predictable range [ (model-null)/(1-null) ]:  0.0006743267
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##    2.956584    -0.004792
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1606  AIC: 1738
## log likelihood:  -802.8277
## Nagelkerke R2:  1.566305e-05
## % pres/err predicted correctly:  -394.2285
## % of predictable range [ (model-null)/(1-null) ]:  7.345e-06
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(posˆ2)	pos	stimlen
preserved ~ CumErr	1526.555	0.0000	1	1	0.1405609	3.338432	NA	-1.075706	NA	NA	NA
preserved ~ CumPres	1705.087	178.5322	0	0	0.0233403	2.463250	0.2081147	NA	NA	NA	NA
preserved ~ stimlen	1711.088	184.5333	0	0	0.0171873	4.601445	NA	NA	NA	NA	-0.2102637
preserved ~ 1	1735.615	209.0602	0	0	0.0000000	2.938210	NA	NA	NA	NA	NA
preserved ~ (I(posˆ2) + pos)	1737.304	210.7498	0	0	0.0022478	3.618822	NA	NA	-0.0244347	0.2077965	NA
preserved ~ pos	1737.560	211.0052	0	0	0.0000157	2.956584	NA	NA	NA	-0.0047924	NA

```
if(DoSimulations){
      BestMEModelFormulaRnd <- BestMEModelFormula
    if(grepl("CumPres",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumPres","RndCumPres",BestMEModelFormulaRnd)
    }else if(grepl("CumErr",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumErr","RndCumErr",BestMEModelFormulaRnd)
    }

    RndModelAIC<-numeric(length=RandomSamples)
    for(rindex in seq(1,RandomSamples)){
    # Shuffle cumulative values
      PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
      PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
      BestModelRnd <- glm(as.formula(BestMEModelFormulaRnd),
                          family="binomial",data=PosDat)
      RndModelAIC[rindex] <- BestModelRnd$aic
    }
    ModelNames<-c(paste0("***",BestMEModelFormula),
                  rep(BestMEModelFormulaRnd,RandomSamples))
    AICValues <- c(BestMEModel$aic,RndModelAIC)
    BestMEModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
    BestMEModelRndDF <- BestMEModelRndDF %>% arrange(AIC)
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random average"),
                                         AIC=c(mean(RndModelAIC))))
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random SD"),
                                         AIC=c(sd(RndModelAIC))))

    write.csv(BestMEModelRndDF,
              paste0(TablesDir,CurPat,"_",CurTask,
                     "_best_main_effects_model_with_random_cum_term.csv"),
              row.names = FALSE)
}
```

```
syll_component_summary <- PosDat %>%
  group_by(syll_component) %>% summarise(MeanPres = mean(preserved),
                                         N = n())
write.csv(syll_component_summary,paste0(TablesDir,CurPat,"_",CurTask,"_syllable_component_summary.csv")
kable(syll_component_summary)
```

syll_component	MeanPres	N
1	0.9653285	548
O	0.9402901	1919
P	0.6000000	30
S	0.9462810	242
V	0.9643865	1418

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.395       -1.181
##
## Degrees of Freedom: 3884 Total (i.e. Null);  3883 Residual
## Null Deviance:       1430
## Residual Deviance: 1244  AIC: 1358
## log likelihood:  -622.1377
## Nagelkerke R2:  0.151583
## % pres/err predicted correctly:  -308.2072
## % of predictable range [ (model-null)/(1-null) ]:  0.1138395
## ***************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.8531      -0.2332
##
## Degrees of Freedom: 3884 Total (i.e. Null);  3883 Residual
## Null Deviance:       1430
## Residual Deviance: 1405  AIC: 1535
## log likelihood:  -702.7031
## Nagelkerke R2:  0.0204834
## % pres/err predicted correctly:  -345.4001
## % of predictable range [ (model-null)/(1-null) ]:  0.007247876
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.6453       0.1594
##
## Degrees of Freedom: 3884 Total (i.e. Null);  3883 Residual
## Null Deviance:       1430
## Residual Deviance: 1415  AIC: 1546
## log likelihood:  -707.28
## Nagelkerke R2:  0.01287115
## % pres/err predicted correctly:  -346.395
## % of predictable range [ (model-null)/(1-null) ]:  0.004396547
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     3.19487     -0.04912
##
## Degrees of Freedom: 3884 Total (i.e. Null);  3883 Residual
## Null Deviance:       1430
## Residual Deviance: 1428  AIC: 1561
## log likelihood:  -714.0011
## Nagelkerke R2:  0.001660182
## % pres/err predicted correctly:  -347.7356
## % of predictable range [ (model-null)/(1-null) ]:  0.0005547431
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##       3.001
##
## Degrees of Freedom: 3884 Total (i.e. Null);  3884 Residual
## Null Deviance:       1430
## Residual Deviance: 1430  AIC: 1562
## log likelihood:  -714.9944
## Nagelkerke R2:  -7.210696e-16
## % pres/err predicted correctly:  -347.9291
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.97872     -0.01481      0.08217
##
## Degrees of Freedom: 3884 Total (i.e. Null);  3882 Residual
## Null Deviance:       1430
## Residual Deviance: 1427  AIC: 1563
## log likelihood:  -713.5086
## Nagelkerke R2:  0.002483045
## % pres/err predicted correctly:  -347.6635
## % of predictable range [ (model-null)/(1-null) ]:  0.0007613137
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1358.415	0.0000	1	1	0.1515830	3.395458	NA	-1.181179	NA	NA	NA
preserved ~ stimlen	1535.197	176.7828	0	0	0.0204834	4.853113	NA	NA	NA	NA	-0.233203
preserved ~ CumPres	1546.359	187.9441	0	0	0.0128712	2.645342	0.1593631	NA	NA	NA	NA
preserved ~ pos	1561.356	202.9411	0	0	0.0016603	2.194866	NA	NA	NA	-0.0491243	NA
preserved ~ 1	1561.548	203.1335	0	0	0.0000000	3.000947	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1562.773	204.3583	0	0	0.0024830	2.978721	NA	NA	-0.0148059	0.0821721	NA

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

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.311       -1.182
##
## Degrees of Freedom: 3336 Total (i.e. Null);  3335 Residual
## Null Deviance:       1274
## Residual Deviance: 1124  AIC: 1208
## log likelihood:  -562.1723
## Nagelkerke R2:  0.1378056
## % pres/err predicted correctly:  -278.4543
## % of predictable range [ (model-null)/(1-null) ]:  0.1053376
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.794       -0.232
##
## Degrees of Freedom: 3336 Total (i.e. Null);  3335 Residual
## Null Deviance:       1274
## Residual Deviance: 1251  AIC: 1346
## log likelihood:  -625.7047
## Nagelkerke R2:  0.02081876
## % pres/err predicted correctly:  -308.9363
## % of predictable range [ (model-null)/(1-null) ]:  0.007750719
## ***************************
## model index:  1
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.6207       0.1803
##
## Degrees of Freedom: 3336 Total (i.e. Null);  3335 Residual
## Null Deviance:       1274
## Residual Deviance: 1260  AIC: 1357
## log likelihood:  -630.1701
## Nagelkerke R2:  0.01242752
## % pres/err predicted correctly:  -310.0244
## % of predictable range [ (model-null)/(1-null) ]:  0.004267065
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.956
##
## Degrees of Freedom: 3336 Total (i.e. Null);  3336 Residual
## Null Deviance:       1274
## Residual Deviance: 1274  AIC: 1369
## log likelihood:  -636.7615
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -311.3573
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     3.13076     -0.04468
##
## Degrees of Freedom: 3336 Total (i.e. Null);  3335 Residual
## Null Deviance:       1274
## Residual Deviance: 1272  AIC: 1369
## log likelihood:  -635.9841
## Nagelkerke R2:  0.001468229
## % pres/err predicted correctly:  -311.1948
## % of predictable range [ (model-null)/(1-null) ]:  0.0005201298
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.97361     -0.01121      0.05384
##
## Degrees of Freedom: 3336 Total (i.e. Null);  3334 Residual
## Null Deviance:       1274
## Residual Deviance: 1271  AIC: 1371
## log likelihood:  -635.7169
## Nagelkerke R2:  0.001972746
## % pres/err predicted correctly:  -311.1547
## % of predictable range [ (model-null)/(1-null) ]:  0.000648452
## **************************
```

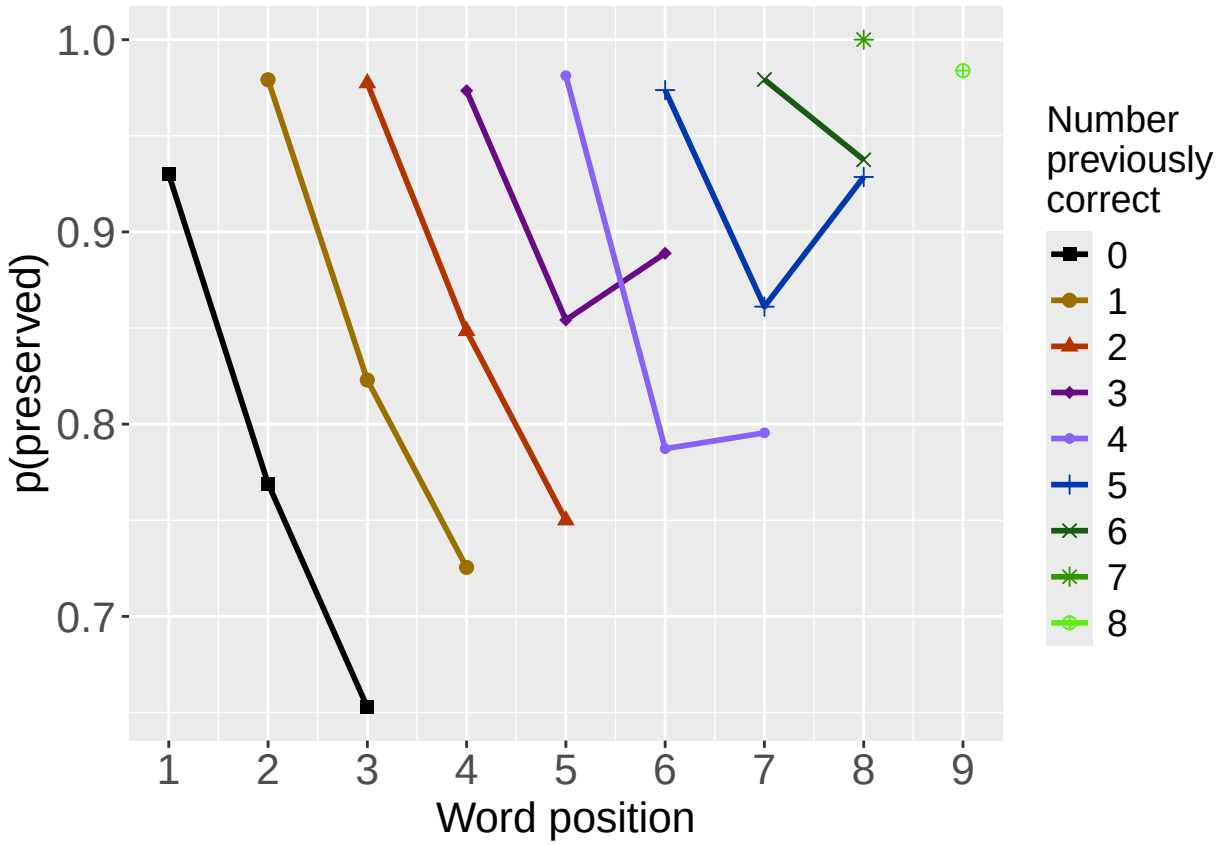
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1208.145	0.0000	1	1	0.137805	3.310542	NA	-1.182372	NA	NA	NA
preserved ~ stimlen	1345.812	137.6677	0	0	0.020818	3.794157	NA	NA	NA	NA	-0.2319994
preserved ~ CumPres	1356.814	148.6697	0	0	0.012427	3.620651	0.1802747	NA	NA	NA	NA
preserved ~ 1	1369.176	161.0317	0	0	0.000000	2.955641	NA	NA	NA	NA	NA
preserved ~ pos	1369.297	161.1525	0	0	0.001468	3.130762	NA	NA	NA	-0.0446759	NA
preserved ~ (I(pos^2) + pos)	1370.964	162.8196	0	0	0.001972	2.973610	NA	NA	-0.0112095	0.0538364	NA

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

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

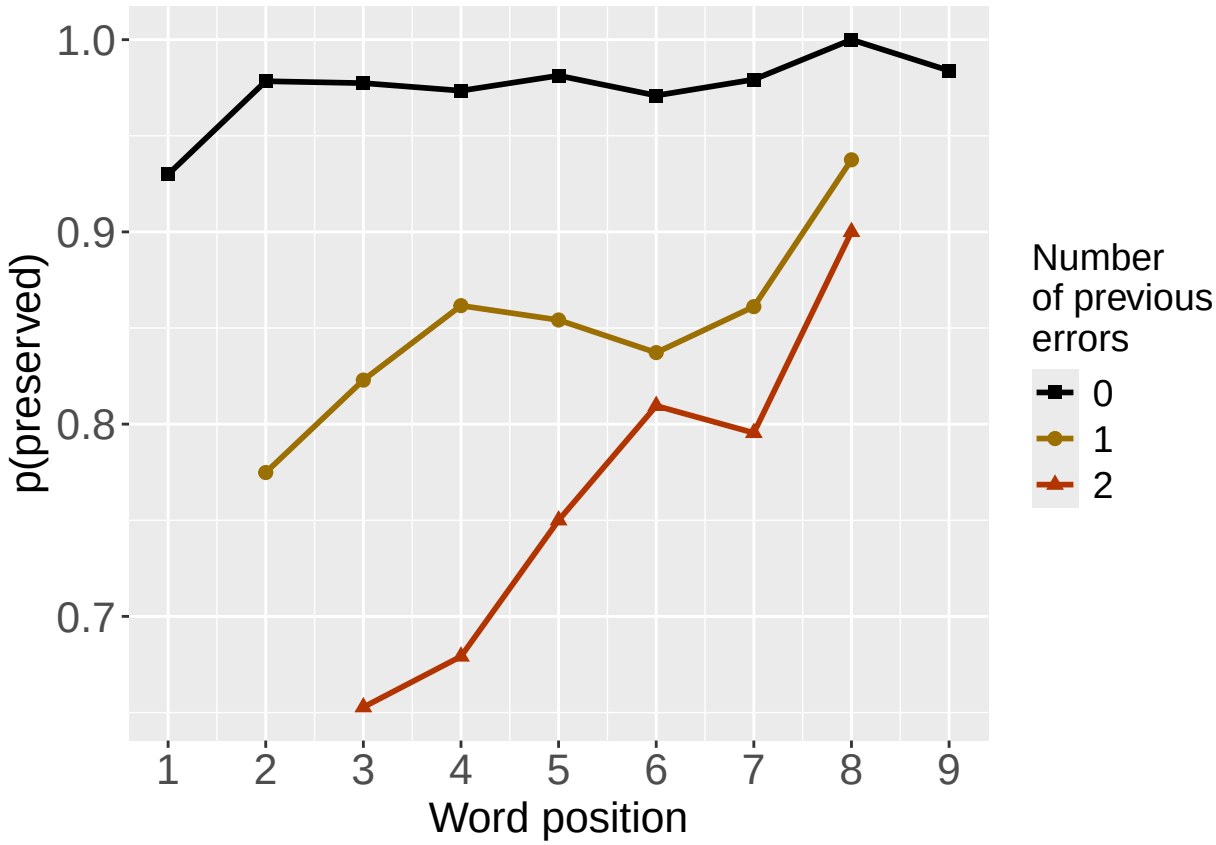
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

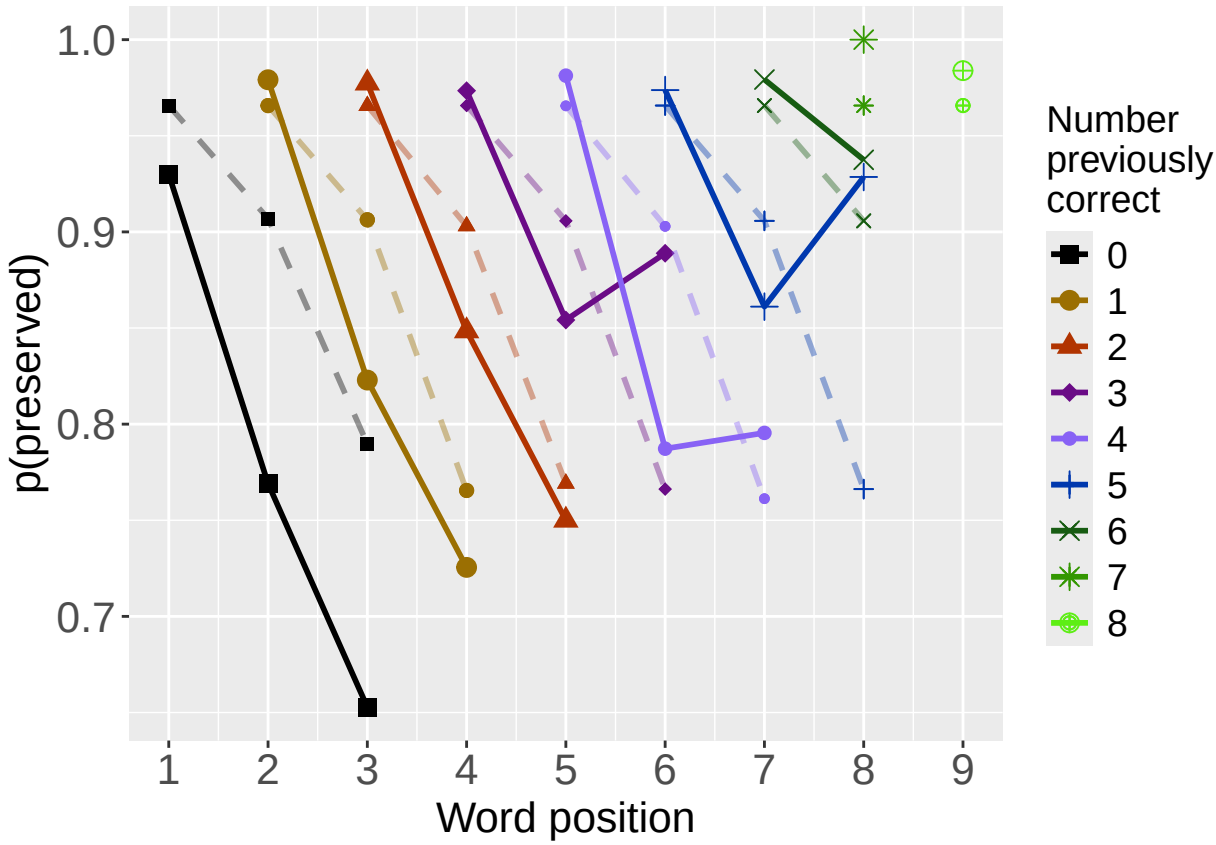
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

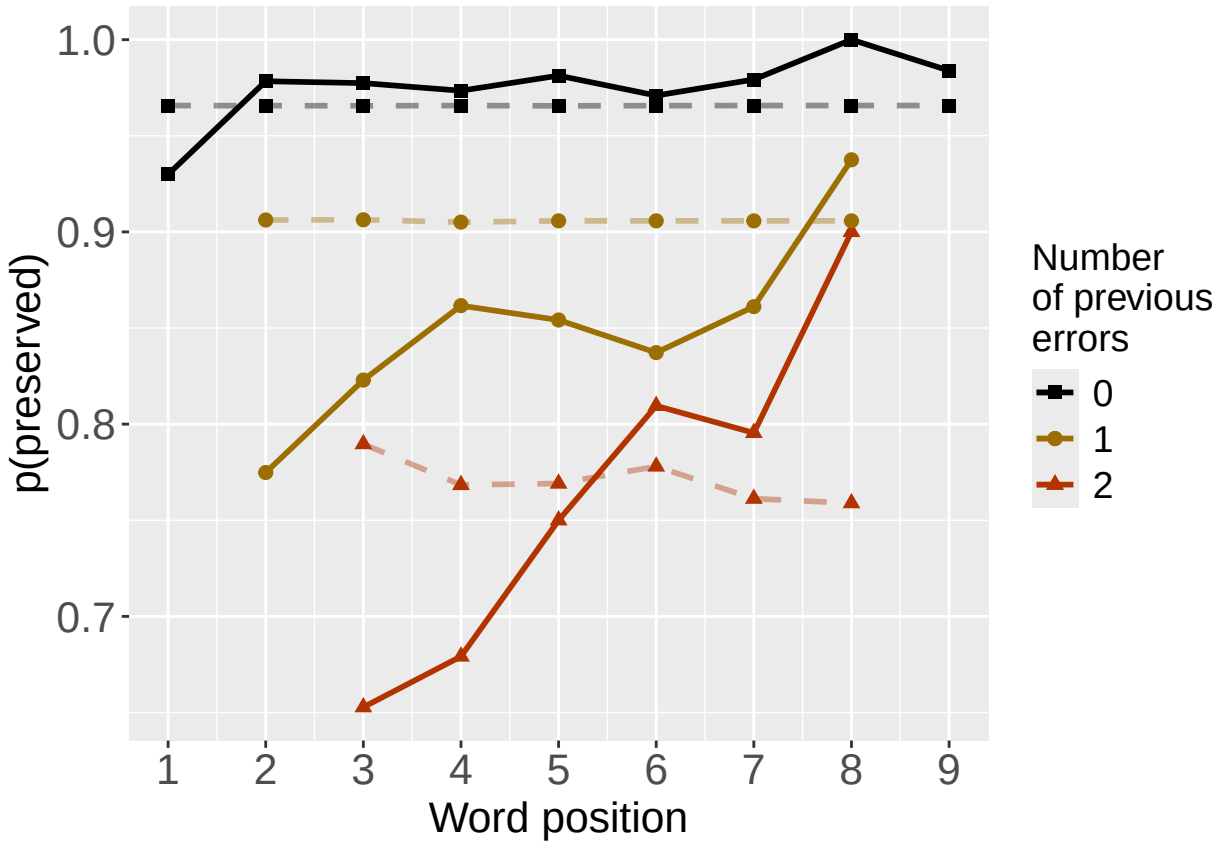
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##    2.635623    -1.322362     0.002405     0.205903
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1384  AIC: 1498
## log likelihood:  -691.8928
## Nagelkerke R2:  0.162226
## % pres/err predicted correctly:  -348.8706
## % of predictable range [ (model-null)/(1-null) ]:  0.1147702
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.338       -1.076
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1414  AIC: 1527
## log likelihood:  -707.0565
## Nagelkerke R2:  0.1405609
## % pres/err predicted correctly:  -353.2909
## % of predictable range [ (model-null)/(1-null) ]:  0.103586
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.61882     -0.02443      0.20780
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1603  AIC: 1737
## log likelihood:  -801.3406
## Nagelkerke R2:  0.00224784
## % pres/err predicted correctly:  -393.9649
## % of predictable range [ (model-null)/(1-null) ]:  0.0006743267
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1498.382	0.00000	1e+00	0.9999992	0.1622260	2.635623	-1.322362	0.0024046	0.2059035

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1526.555	28.17278	8e-07	0.0000008	0.1405609	3.338432	-1.075706	NA	NA
preserved ~ I(posˆ2) + pos	1737.304	238.92259	0e+00	0.0000000	0.0022478	2.618822	NA	-0.0244347	0.2077965

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     4.02895     -1.03826     -0.08977
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1411  AIC: 1524
## log likelihood:  -705.2558
## Nagelkerke R2:  0.143142
## % pres/err predicted correctly:  -352.7974
## % of predictable range [ (model-null)/(1-null) ]:  0.1048346
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.338       -1.076
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1414  AIC: 1527
## log likelihood:  -707.0565
## Nagelkerke R2:  0.1405609
## % pres/err predicted correctly:  -353.2909
## % of predictable range [ (model-null)/(1-null) ]:  0.103586
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.6014      -0.2103
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1583  AIC: 1711
## log likelihood:  -791.3604
## Nagelkerke R2:  0.01718728
## % pres/err predicted correctly:  -391.7077
## % of predictable range [ (model-null)/(1-null) ]:  0.006385407
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	1523.961	0.000000	1.0000000	0.7853332	0.1431420	4.028952	-1.038264	-0.0897719
preserved ~ CumErr	1526.555	2.594043	0.2733448	0.2146668	0.1405609	3.338432	-1.075706	NA
preserved ~ stimlen	1711.088	187.127297	0.0000000	0.0000000	0.0171873	4.601445	NA	-0.2102637

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      2.8317      -1.0955       0.2255
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
```

```
## Null Deviance:       1606
## Residual Deviance: 1384  AIC: 1496
## log likelihood:  -691.9034
## Nagelkerke R2:  0.1622109
## % pres/err predicted correctly:  -348.9362
## % of predictable range [ (model-null)/(1-null) ]:  0.1146042
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.338       -1.076
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1414  AIC: 1527
## log likelihood:  -707.0565
## Nagelkerke R2:  0.1405609
## % pres/err predicted correctly:  -353.2909
## % of predictable range [ (model-null)/(1-null) ]:  0.103586
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.4633       0.2081
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1574  AIC: 1705
## log likelihood:  -787.236
## Nagelkerke R2:  0.02334034
## % pres/err predicted correctly:  -391.0517
## % of predictable range [ (model-null)/(1-null) ]:  0.008045209
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	1496.467	0.00000	1e+00	0.9999997	0.1622109	2.831670	-1.095543	0.2254860
preserved ~ CumErr	1526.555	30.08785	3e-07	0.0000003	0.1405609	3.338432	-1.075706	NA
preserved ~ CumPres	1705.087	208.62007	0e+00	0.0000000	0.0233403	2.463250	NA	0.2081147

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.6062      -1.3210       0.2255
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1384  AIC: 1496
## log likelihood:  -691.9034
## Nagelkerke R2:  0.1622109
## % pres/err predicted correctly:  -348.9362
## % of predictable range [ (model-null)/(1-null) ]:  0.1146042
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.338       -1.076
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1414  AIC: 1527
## log likelihood:  -707.0565
## Nagelkerke R2:  0.1405609
## % pres/err predicted correctly:  -353.2909
## % of predictable range [ (model-null)/(1-null) ]:  0.103586
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##    2.956584    -0.004792
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1606  AIC: 1738
## log likelihood:  -802.8277
## Nagelkerke R2:  1.566305e-05
## % pres/err predicted correctly:  -394.2285
## % of predictable range [ (model-null)/(1-null) ]:  7.345e-06
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	1496.467	0.00000	1e+00	0.9999997	0.1622109	2.606184	-1.321029	0.2254860
preserved ~ CumErr	1526.555	30.08785	3e-07	0.0000003	0.1405609	3.338432	-1.075706	NA
preserved ~ pos	1737.560	241.09301	0e+00	0.0000000	0.0000157	2.956584	NA	-0.0047924

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + pos	1496.467	0.000000	1.000000	0.9999997	0.1622109	2.606184	-1.321029	NA	0.2254860	NA	NA
preserved ~ CumErr + CumPres	1496.467	0.000000	1.000000	0.9999997	0.1622109	2.831670	-1.095543	NA	NA	NA	0.2254860
preserved ~ CumErr + I(pos^2) + pos	1498.382	0.000000	1.000000	0.9999992	0.1622226	2.635623	-1.322362	0.0024046	0.2059035	NA	NA
preserved ~ CumErr + stimlen	1523.960	0.000000	1.000000	0.7853332	0.1431424	2.028952	-1.038264	NA	NA	-0.0897719	NA
preserved ~ CumErr	1526.555	28.172775	0.000000	0.000000	0.1405609	3.338432	-1.075706	NA	NA	NA	NA
preserved ~ CumErr	1526.555	2.594043	0.273344	0.2146668	0.1405609	3.338432	-1.075706	NA	NA	NA	NA
preserved ~ CumErr	1526.555	30.087853	0.000000	0.000000	0.1405609	3.338432	-1.075706	NA	NA	NA	NA
preserved ~ CumErr	1526.555	30.087853	0.000000	0.000000	0.1405609	3.338432	-1.075706	NA	NA	NA	NA
preserved ~ CumPres	1705.087	208.620073	0.000000	0.000000	0.0233402	2.463250	NA	NA	NA	NA	0.2081147
preserved ~ stimlen	1711.088	187.127297	0.000000	0.000000	0.0017187	3.601445	NA	NA	NA	-0.2102637	NA
preserved ~ I(pos^2) + pos	1737.304	238.922593	0.000000	0.000000	0.0022472	2.618822	NA	-0.0244347	0.2077965	NA	NA
preserved ~ pos	1737.560	241.093014	0.000000	0.000000	0.0000157	2.956584	NA	NA	-0.0047924	NA	NA

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen     log_freq
##      3.6606      -1.2755       0.2717      -0.1497       0.1728
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4152 Residual
## Null Deviance:       1606
## Residual Deviance: 1351  AIC: 1462
## log likelihood:  -675.5375
## Nagelkerke R2:  0.1854171
## % pres/err predicted correctly:  -345.0804
## % of predictable range [ (model-null)/(1-null) ]:  0.12436
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq
##      2.6155      -1.2929       0.2401       0.1950
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1360  AIC: 1470
## log likelihood:  -680.0226
## Nagelkerke R2:  0.1790755
## % pres/err predicted correctly:  -346.5514
## % of predictable range [ (model-null)/(1-null) ]:  0.120638
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen
##      3.8907      -1.2913       0.2681      -0.1855
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1369  AIC: 1481
## log likelihood:  -684.5354
## Nagelkerke R2:  0.172681
## % pres/err predicted correctly:  -346.5652
## % of predictable range [ (model-null)/(1-null) ]:  0.1206033
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.6062      -1.3210       0.2255
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1384  AIC: 1496
## log likelihood:  -691.9034
## Nagelkerke R2:  0.1622109
## % pres/err predicted correctly:  -348.9362
## % of predictable range [ (model-null)/(1-null) ]:  0.1146042
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       2.938
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4156 Residual
## Null Deviance:       1606
## Residual Deviance: 1606  AIC: 1736
## log likelihood:  -802.8382
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -394.2314
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos	log_freq	stimlen
preserved ~ CumErr + pos + stimlen + log_freq	1461.999	0.000000	1.000000	0.9857950	0.1854173	1.660639	-1.275518	0.2716866	0.1728222	-0.1496613
preserved ~ CumErr + pos + log_freq	1470.488	8.489075	0.014342	0.0141386	0.1790752	2.615494	-1.292925	0.2401030	0.1949552	NA
preserved ~ CumErr + pos + stimlen	1481.211	19.212005	0.000067	0.0000661	0.1726813	0.890652	-1.291344	0.2681393	NA	-0.1854811
preserved ~ CumErr + pos	1496.467	34.467351	0.000000	0.0000000	0.1622102	2.606184	-1.321029	0.2254860	NA	NA
preserved ~ 1	1735.615	273.615357	0.000000	0.0000000	0.0000000	2.938210	NA	NA	NA	NA

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

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

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

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

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

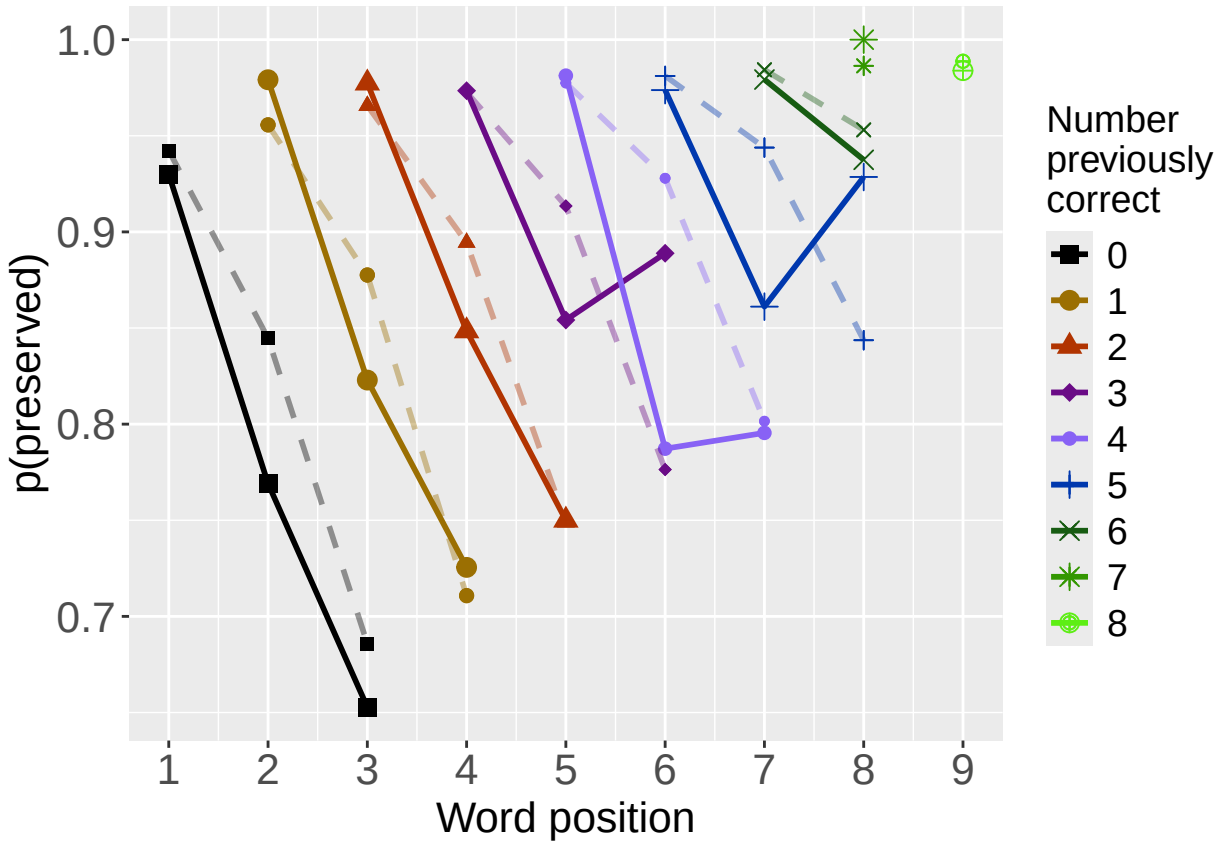
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   1558.8 1667.7
## pos       1   1393.4 1502.3
## log_freq  1   1369.1 1478.0
## stimlen   1   1360.0 1469.0
## <none>        1351.1 1462.0
```

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

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

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

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

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

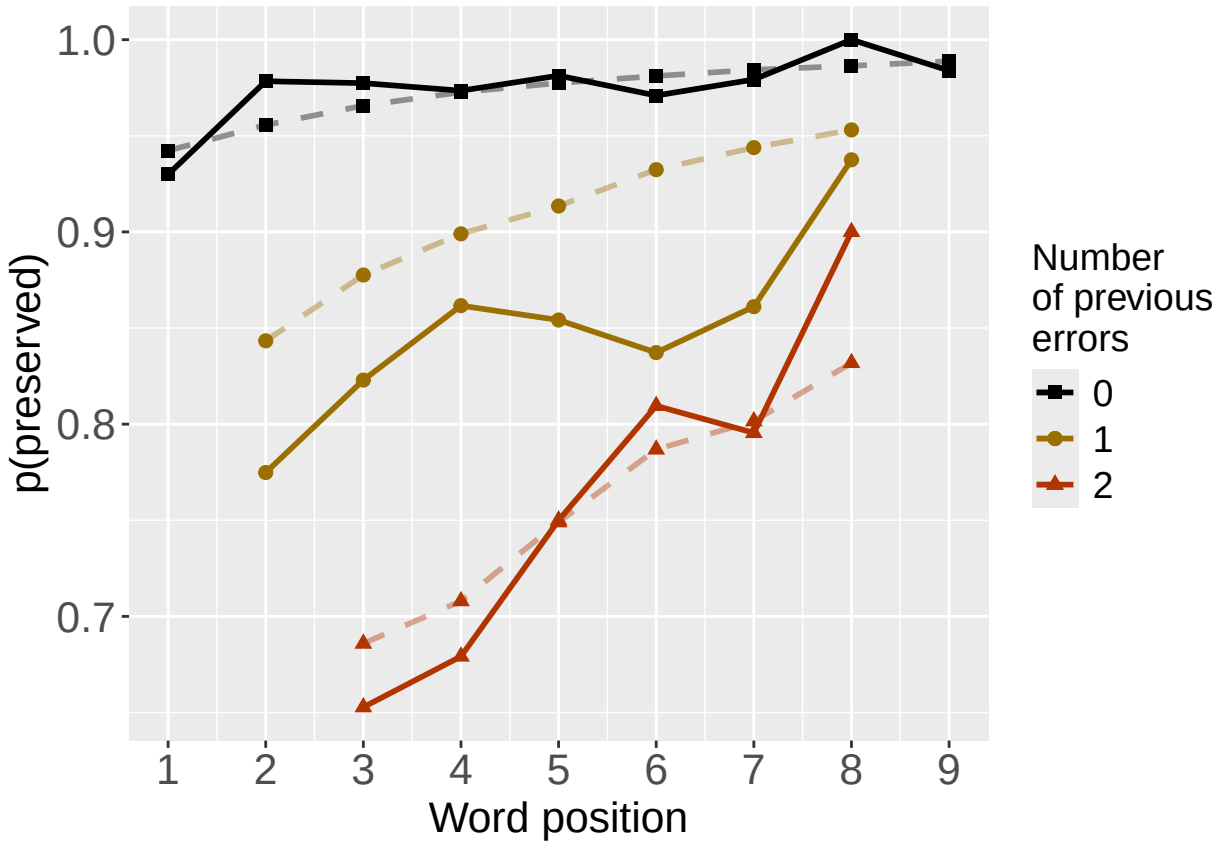
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

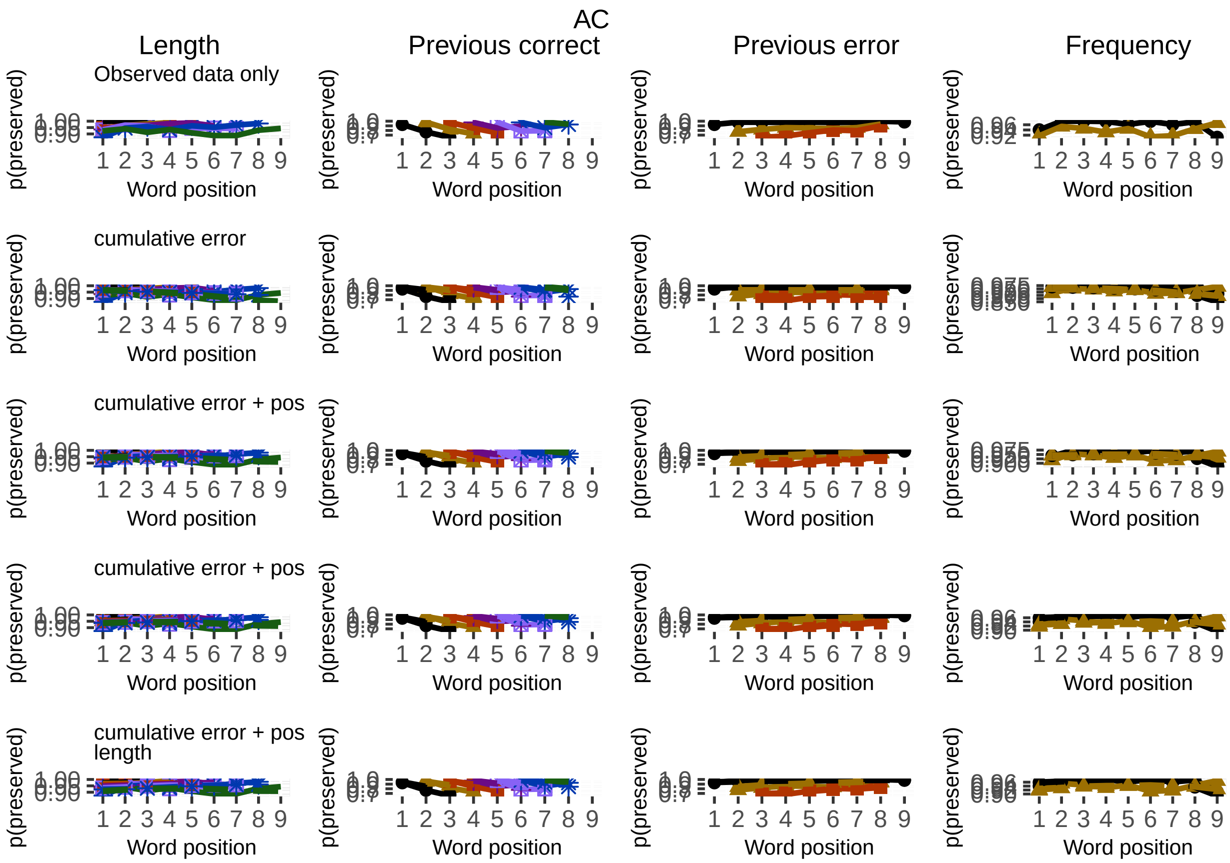
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.338       -1.076
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4155 Residual
## Null Deviance:       1606
## Residual Deviance: 1414  AIC: 1527
## log likelihood:  -707.0565
```

```
## Nagelkerke R2:  0.1405609
## % pres/err predicted correctly:  -353.2909
## % of predictable range [ (model-null)/(1-null) ]:  0.103586
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.6062      -1.3210       0.2255
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4154 Residual
## Null Deviance:       1606
## Residual Deviance: 1384  AIC: 1496
## log likelihood:  -691.9034
## Nagelkerke R2:  0.1622109
## % pres/err predicted correctly:  -348.9362
## % of predictable range [ (model-null)/(1-null) ]:  0.1146042
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq
##      2.6155      -1.2929       0.2401       0.1950
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4153 Residual
## Null Deviance:       1606
## Residual Deviance: 1360  AIC: 1470
## log likelihood:  -680.0226
## Nagelkerke R2:  0.1790755
## % pres/err predicted correctly:  -346.5514
## % of predictable range [ (model-null)/(1-null) ]:  0.120638
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq      stimlen
##      3.6606      -1.2755       0.2717       0.1728      -0.1497
##
## Degrees of Freedom: 4156 Total (i.e. Null);  4152 Residual
## Null Deviance:       1606
## Residual Deviance: 1351  AIC: 1462
## log likelihood:  -675.5375
## Nagelkerke R2:  0.1854171
## % pres/err predicted correctly:  -345.0804
## % of predictable range [ (model-null)/(1-null) ]:  0.12436
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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




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

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

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

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

	CumErr	pos	stimlen	log_freq
McFadden	0.1246990	0.0123794	0.0093762	0.0159894
SquaredCorrelation	0.0503188	0.0048678	0.0038291	0.0064855
Nagelkerke	0.0503188	0.0048678	0.0038291	0.0064855
Estrella	0.0546150	0.0056430	0.0040404	0.0069618

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

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

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

```
##                                                          model deviance
## CumErr + pos + log_freq + stimlen CumErr + pos + log_freq + stimlen 1351.075
## CumErr + pos + log_freq                     CumErr + pos + log_freq 1360.045
## CumErr + pos                                       CumErr + pos 1383.807
## CumErr                                                   CumErr 1414.113
## null                                                       null 1605.676
##                                   deviance_explained percent_explained
## CumErr + pos + log_freq + stimlen           254.6014          15.85633
## CumErr + pos + log_freq                     245.6312          15.29768
## CumErr + pos                                221.8696          13.81783
## CumErr                                      191.5633          11.93038
## null                                          0.0000           0.00000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + pos + log_freq + stimlen                     100.00000               3.523239
## CumErr + pos + log_freq                                96.47676               9.332873
## CumErr + pos                                           87.14389              11.903410
## CumErr                                                 75.24048              75.240477
## null                                                         NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + log_freq + stimlen	1351.075	254.6014
CumErr + pos + log_freq	1360.045	245.6312
CumErr + pos	1383.807	221.8696
CumErr	1414.113	191.5633
null	1605.676	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + log_freq + stimlen	15.85633	100.00000	3.523239
CumErr + pos + log_freq	15.29768	96.47676	9.332873
CumErr + pos	13.81783	87.14389	11.903410
CumErr	11.93038	75.24048	75.240477
null	0.00000	NA	0.000000

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

```
##          Nagelkerke
## CumErr   0.76821254
## pos      0.07431590
## stimlen  0.05845864
## log_freq 0.09901292
```

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

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

```
##                                      model p_accounted_for model_deviance diff_CumErr
## 1                       preserved ~ CumErr       0.7836278       1414.113  0.00000000
## 2                   preserved ~ CumErr+pos       0.8431072       1383.807  0.05947938
## 3          preserved ~ CumErr+pos+log_freq       0.8462273       1360.045  0.06259949
## 4 preserved ~ CumErr+pos+log_freq+stimlen       0.8550067       1351.075  0.07137882
##   diff_CumErr+pos diff_CumErr+pos+log_freq diff_CumErr+pos+log_freq+stimlen
```

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.7836278	1414.113
preserved ~ CumErr+pos	0.8431072	1383.807
preserved ~ CumErr+pos+log_freq	0.8462273	1360.045
preserved ~ CumErr+pos+log_freq+stimlen	0.8550067	1351.075

model	diff_CumErr	diff_CumErr+pos	diff_CumErr+pos+log_freq
preserved ~ CumErr	0.0000000	-0.0594794	-0.0625995
preserved ~ CumErr+pos	0.0594794	0.0000000	-0.0031201
preserved ~ CumErr+pos+log_freq	0.0625995	0.0031201	0.0000000
preserved ~ CumErr+pos+log_freq+stimlen	0.0713788	0.0118994	0.0087793

```
## 1    -0.059479385            -0.062599490                -0.071378819
## 2     0.000000000            -0.003120105                -0.011899434
## 3     0.003120105             0.000000000                -0.008779329
## 4     0.011899434             0.008779329                 0.000000000
```



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

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

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