CA - repetition - Serial position analysis (v5)

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

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

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

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

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

```
# 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	533	33	124	NA	NA	690
2	62	NA	427	95	106	690
3	303	NA	164	208	15	690
4	297	NA	230	67	38	632
5	225	NA	206	70	37	538
6	200	1	131	70	21	423
7	169	NA	99	28	18	314
8	90	NA	49	24	4	167
9	68	NA	2	NA	7	77

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7724638	0.0478261	0.1797101	NA	NA	690
2	0.0898551	NA	0.6188406	0.1376812	0.1536232	690
3	0.4391304	NA	0.2376812	0.3014493	0.0217391	690
4	0.4699367	NA	0.3639241	0.1060127	0.0601266	632
5	0.4182156	NA	0.3828996	0.1301115	0.0687732	538
6	0.4728132	0.0023641	0.3096927	0.1654846	0.0496454	423

pos_factor	O	P	V	1	S	total
7	0.5382166	NA	0.3152866	0.0891720	0.0573248	314
8	0.5389222	NA	0.2934132	0.1437126	0.0239521	167
9	0.8831169	NA	0.0259740	NA	0.0909091	77

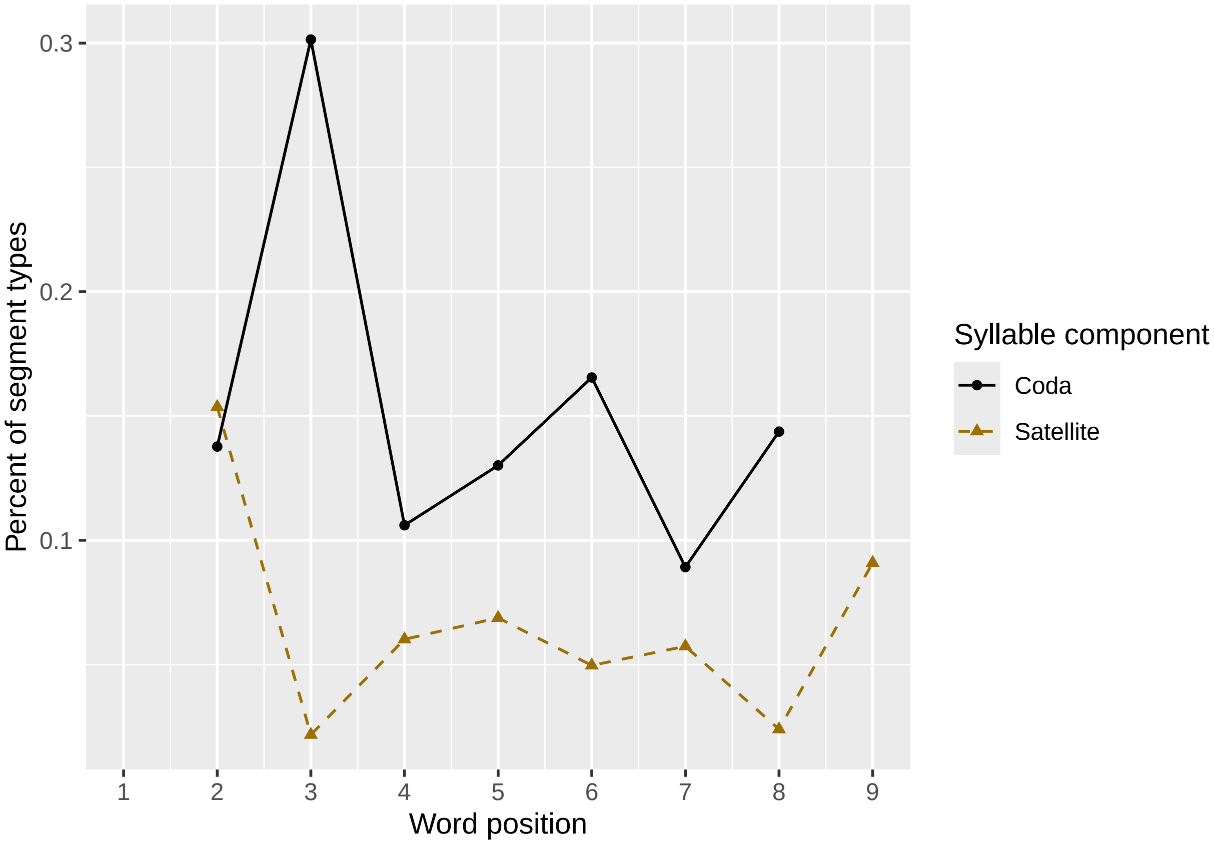
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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



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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.983 1     1     NA     NA     NA     NA     NA     NA    
## 2       5 0.968 0.968 1      0.989 NA     NA     NA     NA     NA    
## 3       6 0.965 0.983 0.974  0.965  0.948 NA     NA     NA     NA    
## 4       7 0.963 0.963 0.972  0.982  0.972  0.982 NA     NA     NA    
## 5       8 0.932 0.918 0.947  0.946  0.964  0.949  0.946 NA     NA    
## 6       9 0.9   0.872 0.894  0.933  0.878  0.933  0.928  0.922 NA    
## 7      10 0.896 0.929 0.883  0.890  0.896  0.929  0.883  0.903  0.883
```

```
# 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.
```

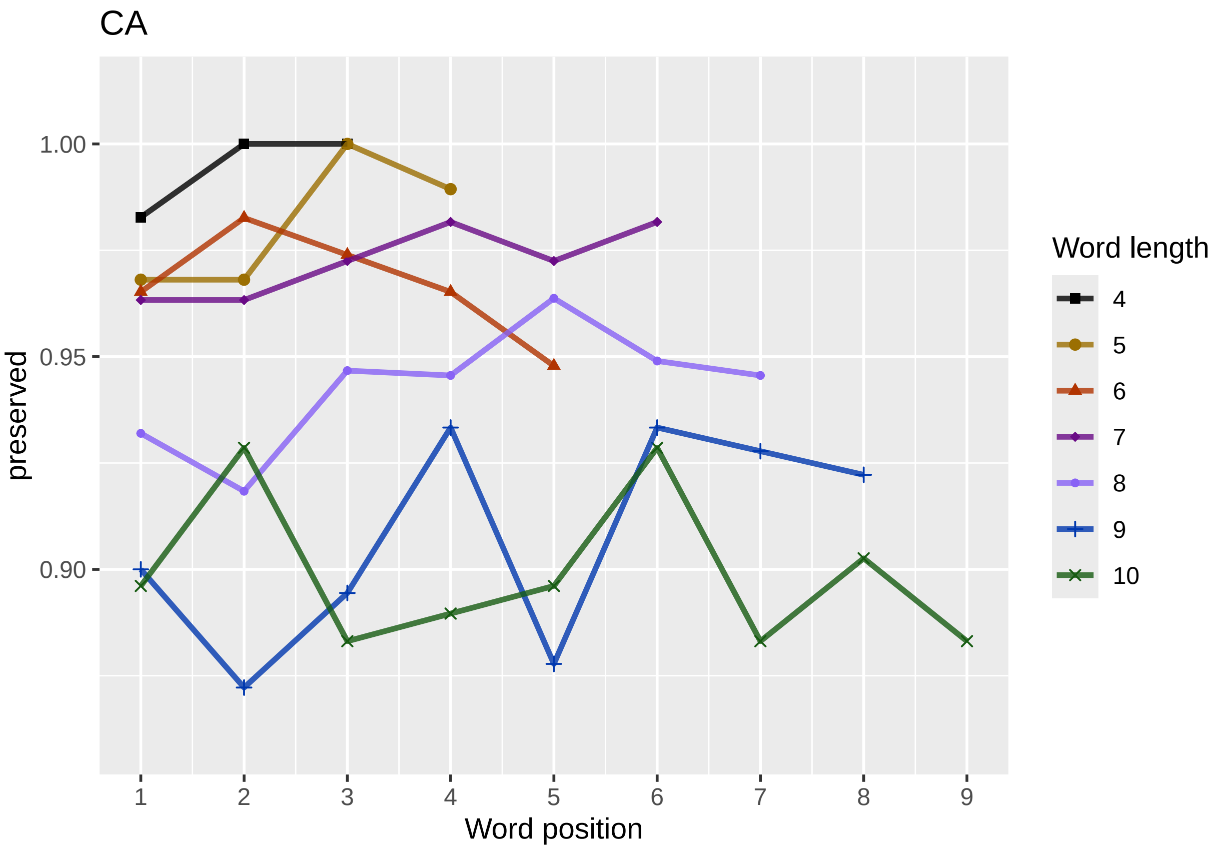
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    94    94    94    94    NA    NA    NA    NA    NA
## 3       6   115   115   115   115   115    NA    NA    NA    NA
## 4       7   109   109   109   109   109   109    NA    NA    NA
## 5       8   147   147   147   147   147   147   147    NA    NA
## 6       9    90    90    90    90    90    90    90    90    NA
## 7      10    77    77    77    77    77    77    77    77    77
```

```
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:  2
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       5.879       -0.380
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1736  AIC: 1819
## log likelihood:  -867.8829
## Nagelkerke R2:  0.05103445
## % pres/err predicted correctly:  -442.1959
## % of predictable range [ (model-null)/(1-null) ]:  0.01873292
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     5.87672     -0.39802      0.03722
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1734  AIC: 1819
## log likelihood:  -867.1529
## Nagelkerke R2:  0.05200785
## % pres/err predicted correctly:  -442.013
## % of predictable range [ (model-null)/(1-null) ]:  0.01913777
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.65776     -0.39143     -0.01146      0.13885
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1734  AIC: 1821
## log likelihood:  -866.7665
## Nagelkerke R2:  0.0525229
## % pres/err predicted correctly:  -441.9553
## % of predictable range [ (model-null)/(1-null) ]:  0.01926549
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     5.40756     -0.34468      0.18385     -0.01633
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1734  AIC: 1821
## log likelihood:  -866.9246
## Nagelkerke R2:  0.05231216
## % pres/err predicted correctly:  -441.9465
## % of predictable range [ (model-null)/(1-null) ]:  0.01928494
## ***************************
## 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)
##         4.746310         -0.286531         -0.082178          0.726088          0.007846
##      stimlen:pos
##        -0.066290
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4215 Residual
## Null Deviance:       1812
## Residual Deviance: 1733  AIC: 1824
## log likelihood:  -866.4973
## Nagelkerke R2:  0.05288172
## % pres/err predicted correctly:  -441.9398
## % of predictable range [ (model-null)/(1-null) ]:  0.01929979
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.62288     -0.02958      0.20082
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1802  AIC: 1895
## log likelihood:  -901.0595
## Nagelkerke R2:  0.00644016
## % pres/err predicted correctly:  -449.4897
## % of predictable range [ (model-null)/(1-null) ]:  0.002583831
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     3.05823     -0.06339
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1807  AIC: 1898
## log likelihood:  -903.6752
## Nagelkerke R2:  0.002894437
## % pres/err predicted correctly:  -450.1118
## % of predictable range [ (model-null)/(1-null) ]:  0.001206399
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.808
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4220 Residual
## Null Deviance:       1812
## Residual Deviance: 1812  AIC: 1900
## log likelihood:  -905.808
## Nagelkerke R2:  6.362956e-16
## % pres/err predicted correctly:  -450.6567
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen	1818.518	0.0000000	1.0000000	0.4042633	0.0510345	4.5878743	-0.3799683	NA	NA	NA	NA
preserved ~ stimlen + pos	1819.027	0.5084192	0.7755292	0.3135181	0.0520073	3.5876719	-0.3980185	0.0372241	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	1820.677	2.1587266	0.3398108	0.1373735	0.0525229	5.657760	-0.3914280	0.1388505	NA	-0.0114629	NA
preserved ~ stimlen * pos	1820.993	2.4749693	0.2901130	0.1172821	0.0523122	5.407560	-0.3446799	0.1838482	-0.0163279	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1823.890	5.3711978	0.0681803	0.0275628	0.0528817	4.746310	-0.2865307	0.7260879	-0.0662902	-0.0821777	0.007846
preserved ~ I(pos^2) + pos	1895.114	76.5956498	0.0000000	0.0000000	0.0064402	2.622876	NA	0.2008221	NA	-0.0295770	NA
preserved ~ pos	1897.597	79.0785635	0.0000000	0.0000000	0.0028944	3.058232	NA	-0.0633864	NA	NA	NA
preserved ~ 1	1900.496	81.9781050	0.0000000	0.0000000	0.0000000	2.807913	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       5.879       -0.380
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1736  AIC: 1819
```

```
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.
```

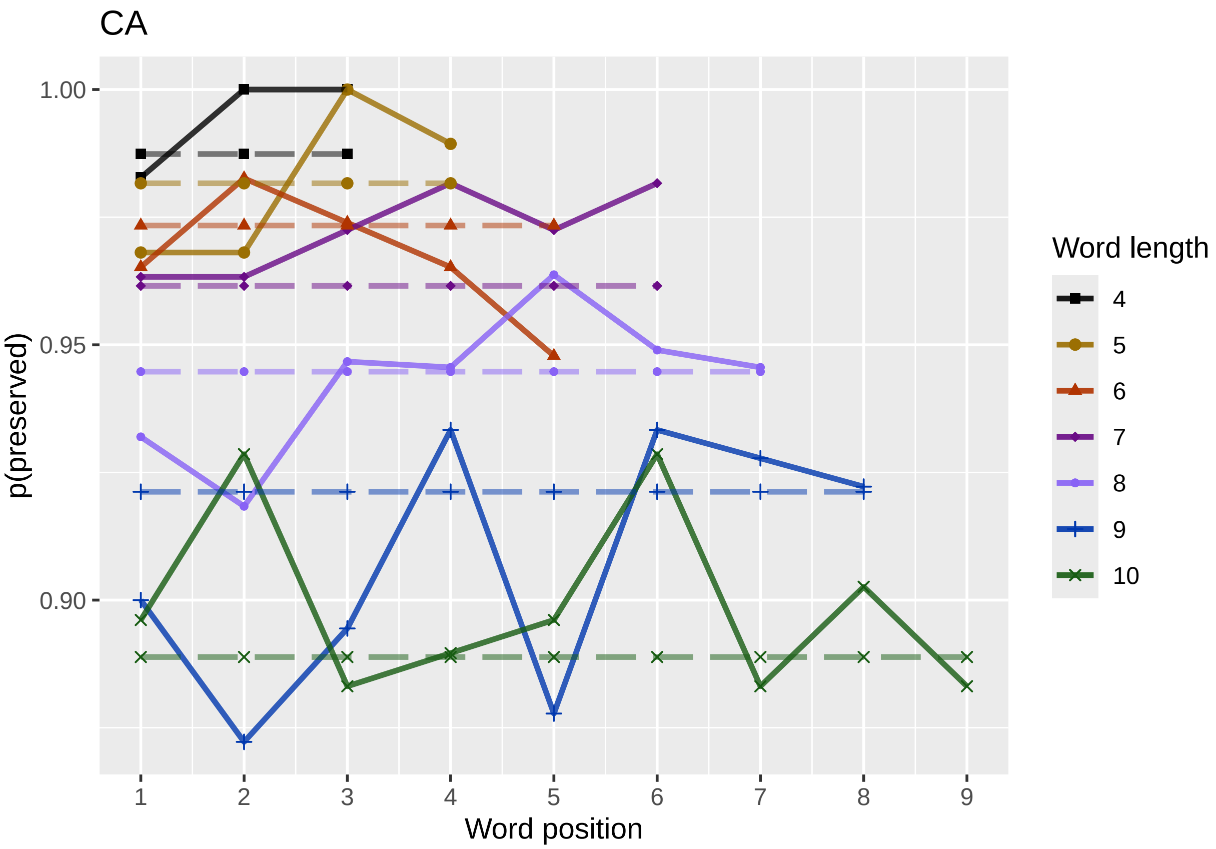
```
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # 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 0.987 0.987 0.987 NA     NA     NA     NA     NA     NA
## 2       5 0.982 0.982 0.982  0.982 NA     NA     NA     NA     NA
## 3       6 0.973 0.973 0.973  0.973  0.973 NA     NA     NA     NA
## 4       7 0.962 0.962 0.962  0.962  0.962  0.962 NA     NA     NA
## 5       8 0.945 0.945 0.945  0.945  0.945  0.945  0.945 NA     NA
## 6       9 0.921 0.921 0.921  0.921  0.921  0.921  0.921  0.921 NA
## 7      10 0.889 0.889 0.889  0.889  0.889  0.889  0.889  0.889  0.889
```

```
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 positionˆ2 influence

```r
# 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
```

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1        8   690
```

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

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

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

```
## [1] "The number of responses with fragments was 8 / 690 = 1.16 percent"
```

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

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

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

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

```
## ****************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen          pos
##    5.70773     -0.38958      0.08766
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4194 Residual
## Null Deviance:       1676
## Residual Deviance: 1609  AIC: 1693
## log likelihood:  -804.649
## Nagelkerke R2:  0.04787457
## % pres/err predicted correctly:  -401.8513
## % of predictable range [ (model-null)/(1-null) ]:  0.01693888
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    5.59867     -0.38645     -0.00618      0.14099
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4193 Residual
## Null Deviance:       1676
## Residual Deviance: 1609  AIC: 1695
## log likelihood:  -804.5536
## Nagelkerke R2:  0.04801052
## % pres/err predicted correctly:  -401.8666
## % of predictable range [ (model-null)/(1-null) ]:  0.01690156
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##   5.562095    -0.372948     0.135134    -0.005317
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4193 Residual
## Null Deviance:       1676
## Residual Deviance: 1609  AIC: 1695
## log likelihood:  -804.6277
## Nagelkerke R2:  0.04790497
## % pres/err predicted correctly:  -401.8585
## % of predictable range [ (model-null)/(1-null) ]:  0.01692118
## ***************************
## 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)
##          4.52513          -0.26691          -0.11737           0.94543           0.01205
##      stimlen:pos
##         -0.08827
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4191 Residual
## Null Deviance:       1676
## Residual Deviance: 1608  AIC: 1697
## log likelihood:  -804.0766
## Nagelkerke R2:  0.04869004
## % pres/err predicted correctly:  -401.8209
## % of predictable range [ (model-null)/(1-null) ]:  0.01701292
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      5.7252      -0.3498
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4195 Residual
## Null Deviance:       1676
## Residual Deviance: 1616  AIC: 1698
## log likelihood:  -808.2438
## Nagelkerke R2:  0.04274904
## % pres/err predicted correctly:  -402.8047
## % of predictable range [ (model-null)/(1-null) ]:  0.0146123
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.911
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4196 Residual
## Null Deviance:       1676
## Residual Deviance: 1676  AIC: 1763
## log likelihood:  -837.9884
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -408.7927
## % 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
##     2.60293     -0.02422      0.20341
```

```
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4194 Residual
## Null Deviance:       1676
## Residual Deviance: 1673  AIC: 1764
## log likelihood:  -836.4878
## Nagelkerke R2:  0.002171204
## % pres/err predicted correctly:  -408.5301
## % of predictable range [ (model-null)/(1-null) ]:  0.0006409564
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##    2.935347    -0.006441
##
## Degrees of Freedom: 4196 Total (i.e. Null);  4195 Residual
## Null Deviance:       1676
## Residual Deviance: 1676  AIC: 1764
## log likelihood:  -837.9691
## Nagelkerke R2:  2.793298e-05
## % pres/err predicted correctly:  -408.7858
## % of predictable range [ (model-null)/(1-null) ]:  1.697755e-05
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	1692.740	0.000000	1.000000	0.5297410	0.0047874	5.6707729	-0.3895784	0.0876636	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	1694.796	2.055572	0.3577983	0.1895408	0.0048010	5.5598672	-0.3864523	0.1409921	NA	-0.0061801	NA
preserved ~ stimlen * pos	1694.843	2.103070	0.3494001	0.1850924	0.0047905	5.562094	-0.3729476	0.1351338	-0.0053171	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)	1697.154	4.414296	0.1100140	0.0582790	0.0486940	4.525129	-0.2669080	0.9454291	-0.0882709	-0.1173697	0.0120461
preserved ~ stimlen	1698.044	5.304330	0.0704984	0.0373460	0.0427495	5.725201	-0.3498182	NA	NA	NA	NA
preserved ~ 1	1762.583	69.842936	0.0000000	0.0000000	0.0000000	2.910761	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1764.248	71.507558	0.0000000	0.0000000	0.0021712	2.602932	NA	0.2034074	NA	-0.0242197	NA
preserved ~ pos	1764.490	71.750124	0.0000000	0.0000000	0.0000279	2.935347	NA	-0.0064413	NA	NA	NA

```r
# plot no fragment data

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

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

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

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

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

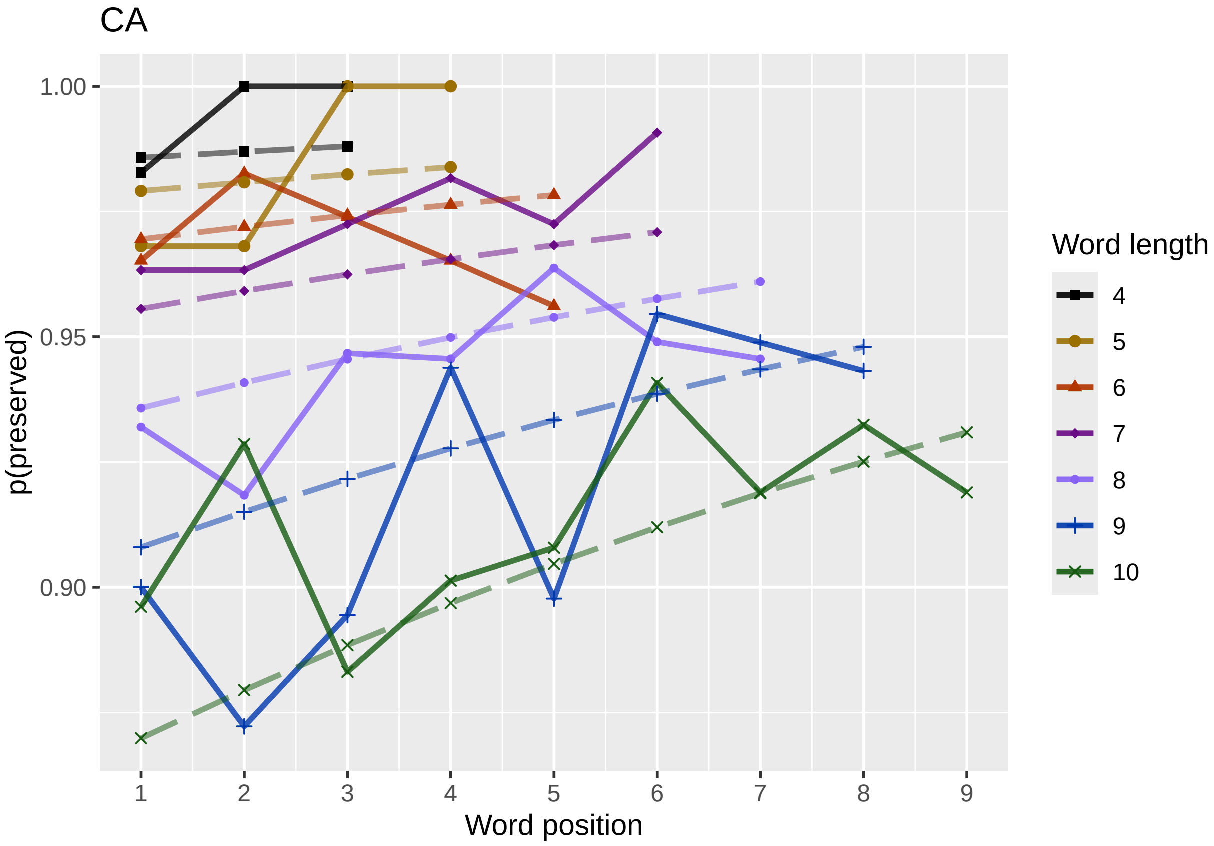
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.986 0.987 0.988 NA     NA     NA     NA     NA     NA
## 2       5 0.979 0.981 0.982  0.984 NA     NA     NA     NA     NA
## 3       6 0.969 0.972 0.974  0.976  0.978 NA     NA     NA     NA
## 4       7 0.956 0.959 0.962  0.965  0.968  0.971 NA     NA     NA
## 5       8 0.936 0.941 0.946  0.950  0.954  0.958  0.961 NA     NA
## 6       9 0.908 0.915 0.922  0.928  0.933  0.939  0.943  0.948 NA
## 7      10 0.870 0.879 0.888  0.897  0.905  0.912  0.919  0.925  0.931
```

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

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

```
                                          )
```

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

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"no_fragments_percent_preserved_by_length_pos_wfit.png"),plot=n
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
```

```r
# take the difference between each value (differences between position proportion correct) **NOTE** pro
# average the difference in probabilities

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

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.0164196
```

```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] NaN
```

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

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

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

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

```
## [1] "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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      5.5194      -0.3256       0.1886
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1709  AIC: 1794
## log likelihood:  -854.4694
## Nagelkerke R2:  0.06886617
## % pres/err predicted correctly:  -438.9687
## % of predictable range [ (model-null)/(1-null) ]:  0.025878
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     5.51728     -0.34380      0.03757      0.18873
```

```
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1707  AIC: 1794
## log likelihood:  -853.7325
## Nagelkerke R2:  0.06984246
## % pres/err predicted correctly:  -438.7373
## % of predictable range [ (model-null)/(1-null) ]:  0.02639052
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          5.59994          -0.33659           0.41739          -0.02694
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1708  AIC: 1795
## log likelihood:  -853.9175
## Nagelkerke R2:  0.06959738
## % pres/err predicted correctly:  -439.1137
## % of predictable range [ (model-null)/(1-null) ]:  0.02555697
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          5.59770          -0.35474           0.41726           0.03751          -0.02692
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4216 Residual
## Null Deviance:       1812
## Residual Deviance: 1706  AIC: 1795
## log likelihood:  -853.1819
## Nagelkerke R2:  0.07057173
## % pres/err predicted correctly:  -438.8808
## % of predictable range [ (model-null)/(1-null) ]:  0.02607263
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      5.53256      -0.34045       0.02637       0.23808      -0.01201
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4216 Residual
## Null Deviance:       1812
## Residual Deviance: 1707  AIC: 1795
## log likelihood:  -853.4497
## Nagelkerke R2:  0.07021705
## % pres/err predicted correctly:  -438.6611
## % of predictable range [ (model-null)/(1-null) ]:  0.02655912
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     5.28482     -0.33666     -0.01213      0.14499      0.18906
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4216 Residual
## Null Deviance:       1812
## Residual Deviance: 1707  AIC: 1796
## log likelihood:  -853.3055
## Nagelkerke R2:  0.07040808
## % pres/err predicted correctly:  -438.675
## % of predictable range [ (model-null)/(1-null) ]:  0.02652846
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          5.35688          -0.34751           0.42406          -0.01265           0.14948
## stimlen:log_freq
##         -0.02768
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4215 Residual
## Null Deviance:       1812
## Residual Deviance: 1705  AIC: 1797
## log likelihood:  -852.7183
## Nagelkerke R2:  0.07118565
## % pres/err predicted correctly:  -438.8195
## % of predictable range [ (model-null)/(1-null) ]:  0.02620852
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         5.598338         -0.351379          0.418206          0.030464         -0.023276
##     log_freq:pos
##        -0.007757
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4215 Residual
## Null Deviance:       1812
## Residual Deviance: 1706  AIC: 1797
## log likelihood:  -853.0745
## Nagelkerke R2:  0.07071403
## % pres/err predicted correctly:  -438.8105
## % of predictable range [ (model-null)/(1-null) ]:  0.02622836
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          5.241984          -0.331922          -0.015572           0.163695           0.192351
## I(pos^2):log_freq       pos:log_freq
##         -0.003343           0.017393
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4214 Residual
## Null Deviance:       1812
## Residual Deviance: 1706  AIC: 1799
## log likelihood:  -852.8584
## Nagelkerke R2:  0.07100019
## % pres/err predicted correctly:  -438.6093
## % of predictable range [ (model-null)/(1-null) ]:  0.02667382
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##           5.31386           -0.34248            0.37488           -0.01497            0.16236
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##          -0.02206           -0.00249            0.01390
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4213 Residual
## Null Deviance:       1812
## Residual Deviance: 1705  AIC: 1800
## log likelihood:  -852.5289
## Nagelkerke R2:  0.07143645
## % pres/err predicted correctly:  -438.7476
## % of predictable range [ (model-null)/(1-null) ]:  0.02636771
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       5.879       -0.380
##
```

```
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1736  AIC: 1819
## log likelihood:  -867.8829
## Nagelkerke R2:  0.05103445
## % pres/err predicted correctly:  -442.1959
## % of predictable range [ (model-null)/(1-null) ]:  0.01873292
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     5.87672     -0.39802      0.03722
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1734  AIC: 1819
## log likelihood:  -867.1529
## Nagelkerke R2:  0.05200785
## % pres/err predicted correctly:  -442.013
## % of predictable range [ (model-null)/(1-null) ]:  0.01913777
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.65776     -0.39143     -0.01146      0.13885
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1734  AIC: 1821
## log likelihood:  -866.7665
## Nagelkerke R2:  0.0525229
## % pres/err predicted correctly:  -441.9553
## % of predictable range [ (model-null)/(1-null) ]:  0.01926549
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     5.40756     -0.34468      0.18385     -0.01633
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1734  AIC: 1821
## log likelihood:  -866.9246
## Nagelkerke R2:  0.05231216
## % pres/err predicted correctly:  -441.9465
## % of predictable range [ (model-null)/(1-null) ]:  0.01928494
## ***************************
## 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)
##         4.746310         -0.286531         -0.082178          0.726088          0.007846
##      stimlen:pos
##        -0.066290
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4215 Residual
## Null Deviance:       1812
## Residual Deviance: 1733  AIC: 1824
## log likelihood:  -866.4973
## Nagelkerke R2:  0.05288172
## % pres/err predicted correctly:  -441.9398
## % of predictable range [ (model-null)/(1-null) ]:  0.01929979
## ***************************
## 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
##          2.685004          -0.031646           0.220687           0.283135          -0.004512
##      pos:log_freq
##          0.015560
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4215 Residual
## Null Deviance:       1812
## Residual Deviance: 1750  AIC: 1847
## log likelihood:  -874.95
## Nagelkerke R2:  0.04159396
## % pres/err predicted correctly:  -443.7868
## % of predictable range [ (model-null)/(1-null) ]:  0.01521053
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      3.14821      -0.06114       0.34894      -0.02432
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1755  AIC: 1847
## log likelihood:  -877.4601
## Nagelkerke R2:  0.03823327
## % pres/err predicted correctly:  -444.2941
## % of predictable range [ (model-null)/(1-null) ]:  0.01408719
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     3.06333     -0.03855      0.25059
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1757  AIC: 1848
## log likelihood:  -878.5545
## Nagelkerke R2:  0.03676665
## % pres/err predicted correctly:  -444.6151
## % of predictable range [ (model-null)/(1-null) ]:  0.01337661
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.62288     -0.02958      0.20082
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1802  AIC: 1895
## log likelihood:  -901.0595
## Nagelkerke R2:  0.00644016
## % pres/err predicted correctly:  -449.4897
## % of predictable range [ (model-null)/(1-null) ]:  0.002583831
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     3.05823     -0.06339
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1807  AIC: 1898
## log likelihood:  -903.6752
## Nagelkerke R2:  0.002894437
## % pres/err predicted correctly:  -450.1118
## % of predictable range [ (model-null)/(1-null) ]:  0.001206399
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.808
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4220 Residual
## Null Deviance:       1812
## Residual Deviance: 1812  AIC: 1900
## log likelihood:  -905.808
## Nagelkerke R2:  6.362956e-16
## % pres/err predicted correctly:  -450.6567
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC DeltaAICexpAICwtNagR2	(Intercept)	stimlen log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	pos:I(pos^2)	log_freq:I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + log_freq	1793.0.53001.000000.2366256885612941	-0.3255938	0.1886369	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	1794.1.48435781001848.0698.425727	-0.3437960	0.1887346	0.0375731	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	1794.5.64100.334012980.5162959799942	-0.3365871	0.4173860 -0.0269421	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	1795.0.6108083945824703078577698	-0.3547431	0.4172558 -0.0269166	0.0375079	NA	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(posˆ2)	I(posˆ2):log_freq	log_freq:I(posˆ2)	stimlen:pos	stimlen:I(posˆ2)
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3404463	0.2380836	NA	0.0263746	-0.0120099	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(posˆ2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3366569	0.1890624	NA	0.1449901	NA	NA	-0.0121284	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(posˆ2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3475089	0.4240622	-0.0276788	0.1494824	NA	NA	-0.0126480	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3513788	0.4182057	-0.0232759	0.0304643	NA	-0.0077568	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(posˆ2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3319224	0.1923509	NA	0.1636945	0.0157393	NA	-0.0155705	-0.0233425	NA	NA	NA
preserved ~ stimlen * log_freq + (I(posˆ2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3424756	0.3748751	-0.0220571	0.1623563	NA	0.0139006	-0.0149652	NA	-0.0024897	NA	NA
preserved ~ stimlen	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3799683	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3980185	NA	NA	0.0372241	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(posˆ2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3914280	NA	NA	0.1388505	NA	NA	-0.0114629	NA	NA	NA	NA
preserved ~ stimlen * pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3446799	NA	NA	0.1838482	NA	NA	NA	NA	NA	-0.0163279	NA
preserved ~ stimlen * (I(posˆ2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2865307	NA	NA	0.7260879	NA	NA	-0.0821777	NA	NA	-0.0662902	0.007846
preserved ~ (I(posˆ2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.2831349	NA	0.2206875	0.0755604	NA	-0.0316406	-0.0045125	NA	NA	NA
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.3489100	NA	-0.0611352	-0.0543211	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAICexpAICwNagR2(Intercept)stimlenlog_freqstimlen:poslog_pos:log_freqlog_freq:I(pos^2)I(pos^2)log_freq:stimlen:I(pos^2)										
preserved ~ pos + log_freq	1847.944291000400000000363766732NA	0.2505871	NA	-0.0385538	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1895.1011.46069480000000062462287NA	NA	NA	0.2008221	NA	NA	-0.0295770	NA	NA	NA	NA
preserved ~ pos	1897.5937.94036085000000028958232NA	NA	NA	-0.0633864	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	1900.4967.84031500000000002807913NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      5.5194      -0.3256       0.1886
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1709  AIC: 1794
```

```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.
```

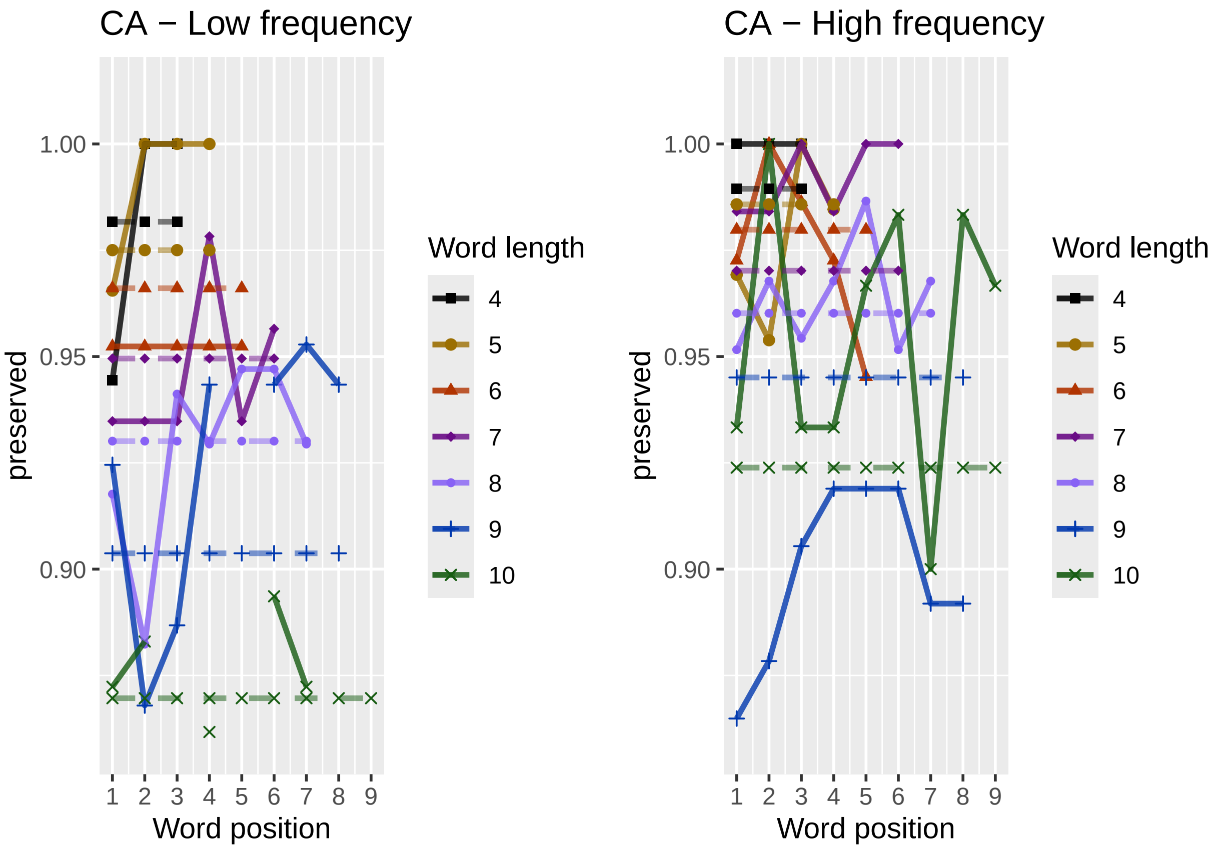
```r
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

```
## `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)
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

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



```r
# 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
##       3.365       -1.241
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1449  AIC: 1517
## log likelihood:  -724.6516
## Nagelkerke R2:  0.2357118
## % pres/err predicted correctly:  -355.6997
## % of predictable range [ (model-null)/(1-null) ]:  0.2102416
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       5.879       -0.380
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1736  AIC: 1819
## log likelihood:  -867.8829
## Nagelkerke R2:  0.05103445
## % pres/err predicted correctly:  -442.1959
## % of predictable range [ (model-null)/(1-null) ]:  0.01873292
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.2023       0.2828
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1751  AIC: 1841
## log likelihood:  -875.4459
## Nagelkerke R2:  0.0409303
## % pres/err predicted correctly:  -443.8285
## % of predictable range [ (model-null)/(1-null) ]:  0.01511815
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.62288     -0.02958      0.20082
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1802  AIC: 1895
## log likelihood:  -901.0595
## Nagelkerke R2:  0.00644016
## % pres/err predicted correctly:  -449.4897
## % of predictable range [ (model-null)/(1-null) ]:  0.002583831
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     3.05823     -0.06339
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1807  AIC: 1898
## log likelihood:  -903.6752
## Nagelkerke R2:  0.002894437
## % pres/err predicted correctly:  -450.1118
## % of predictable range [ (model-null)/(1-null) ]:  0.001206399
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.808
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4220 Residual
## Null Deviance:       1812
## Residual Deviance: 1812  AIC: 1900
## log likelihood:  -905.808
## Nagelkerke R2:  6.362956e-16
## % pres/err predicted correctly:  -450.6567
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1516.649	0.0000	1	1	0.2357111	3.364632	NA	-1.241392	NA	NA	NA
preserved ~ stimlen	1818.518	301.8690	0	0	0.0510345	5.878743	NA	NA	NA	NA	-0.3799683
preserved ~ CumPres	1841.236	324.5863	0	0	0.0409303	2.202342	0.282753	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1895.114	378.4647	0	0	0.0064402	2.622876	NA	NA	-0.029577	0.2008221	NA
preserved ~ pos	1897.597	380.9476	0	0	0.0028944	1.058232	NA	NA	NA	-0.0633864	NA
preserved ~ 1	1900.497	383.8471	0	0	0.0000000	2.807913	NA	NA	NA	NA	NA

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

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

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

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

syll_component	MeanPres	N
1	0.9564057	562
O	0.9414484	1947
P	0.9411765	34
S	0.8943089	246
V	0.9485568	1432

```
#  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
##       3.445       -1.408
##
## Degrees of Freedom: 3940 Total (i.e. Null);  3939 Residual
## Null Deviance:       1624
## Residual Deviance: 1269  AIC: 1334
## log likelihood:  -634.7391
## Nagelkerke R2:  0.2545295
## % pres/err predicted correctly:  -307.7168
## % of predictable range [ (model-null)/(1-null) ]:  0.2288299
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      6.0874      -0.3974
##
## Degrees of Freedom: 3940 Total (i.e. Null);  3939 Residual
## Null Deviance:       1624
## Residual Deviance: 1551  AIC: 1627
## log likelihood:  -775.5737
## Nagelkerke R2:  0.05397685
## % pres/err predicted correctly:  -391.5241
## % of predictable range [ (model-null)/(1-null) ]:  0.01948051
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.2671       0.3011
##
## Degrees of Freedom: 3940 Total (i.e. Null);  3939 Residual
## Null Deviance:       1624
## Residual Deviance: 1569  AIC: 1653
## log likelihood:  -784.3443
## Nagelkerke R2:  0.04100678
## % pres/err predicted correctly:  -393.3706
## % of predictable range [ (model-null)/(1-null) ]:  0.01486797
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.75398     -0.02836      0.17693
##
```

```
## Degrees of Freedom: 3940 Total (i.e. Null);  3938 Residual
## Null Deviance:       1624
## Residual Deviance: 1614  AIC: 1699
## log likelihood:  -806.7674
## Nagelkerke R2:  0.007583527
## % pres/err predicted correctly:  -398.1505
## % of predictable range [ (model-null)/(1-null) ]:  0.002927939
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     3.17934     -0.07814
##
## Degrees of Freedom: 3940 Total (i.e. Null);  3939 Residual
## Null Deviance:       1624
## Residual Deviance: 1618  AIC: 1701
## log likelihood:  -808.927
## Nagelkerke R2:  0.004344355
## % pres/err predicted correctly:  -398.6334
## % of predictable range [ (model-null)/(1-null) ]:  0.001721587
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.866
##
## Degrees of Freedom: 3940 Total (i.e. Null);  3940 Residual
## Null Deviance:       1624
## Residual Deviance: 1624  AIC: 1705
## log likelihood:  -811.8197
## Nagelkerke R2:  -6.575848e-16
## % pres/err predicted correctly:  -399.3226
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1333.982	0.0000	1	1	0.2545295	3.445148	NA	-1.407589	NA	NA	NA
preserved ~ stimlen	1627.059	293.0773	0	0	0.0539766	0.087354	NA	NA	NA	NA	-0.3974384

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	1652.936	318.9543	0	0	0.0410068	2.267104	0.3011392	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1699.213	365.2310	0	0	0.0075835	2.753977	NA	NA	-0.0283623	0.1769279	NA
preserved ~ pos	1700.761	366.7794	0	0	0.0043443	3.179338	NA	NA	NA	-0.0781402	NA
preserved ~ 1	1705.166	371.1847	0	0	0.0000000	2.866479	NA	NA	NA	NA	NA

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.359       -1.526
##
## Degrees of Freedom: 3378 Total (i.e. Null);  3377 Residual
## Null Deviance:       1428
## Residual Deviance: 1151  AIC: 1196
## log likelihood:  -575.6488
## Nagelkerke R2:  0.2279368
## % pres/err predicted correctly:  -279.9868
## % of predictable range [ (model-null)/(1-null) ]:  0.2052686
## ****************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen
##      5.9315      -0.3837
##
## Degrees of Freedom: 3378 Total (i.e. Null);  3377 Residual
## Null Deviance:       1428
## Residual Deviance: 1367  AIC: 1424
## log likelihood:  -683.3929
## Nagelkerke R2:  0.05186153
## % pres/err predicted correctly:  -345.8746
## % of predictable range [ (model-null)/(1-null) ]:  0.01891425
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.2830       0.3305
##
## Degrees of Freedom: 3378 Total (i.e. Null);  3377 Residual
## Null Deviance:       1428
## Residual Deviance: 1384  AIC: 1447
## log likelihood:  -691.7681
## Nagelkerke R2:  0.03769859
## % pres/err predicted correctly:  -347.479
## % of predictable range [ (model-null)/(1-null) ]:  0.01437644
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.67366     -0.02753      0.18583
##
## Degrees of Freedom: 3378 Total (i.e. Null);  3376 Residual
## Null Deviance:       1428
## Residual Deviance: 1421  AIC: 1485
## log likelihood:  -710.469
## Nagelkerke R2:  0.005819953
## % pres/err predicted correctly:  -351.7417
## % of predictable range [ (model-null)/(1-null) ]:  0.002320165
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     3.06333     -0.05824
##
## Degrees of Freedom: 3378 Total (i.e. Null);  3377 Residual
## Null Deviance:       1428
## Residual Deviance: 1425  AIC: 1486
## log likelihood:  -712.3648
## Nagelkerke R2:  0.002568484
## % pres/err predicted correctly:  -352.1697
## % of predictable range [ (model-null)/(1-null) ]:  0.001109426
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.834
##
## Degrees of Freedom: 3378 Total (i.e. Null);  3378 Residual
## Null Deviance:       1428
## Residual Deviance: 1428  AIC: 1488
## log likelihood:  -713.8609
## Nagelkerke R2:  3.221661e-16
## % pres/err predicted correctly:  -352.562
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

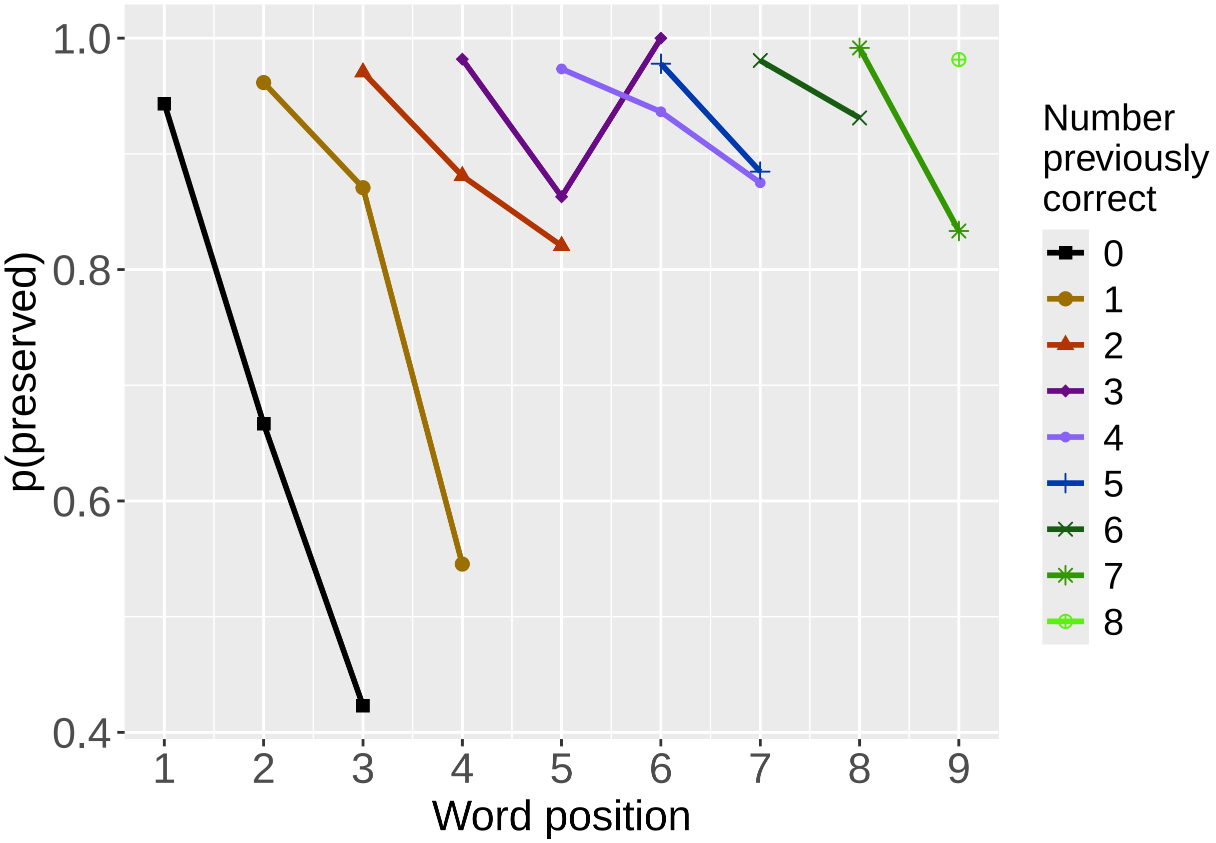
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1196.477	0.0000	1	1	0.2279368	3.358895	NA	- 1.52648	NA	NA	NA
preserved ~ stimlen	1423.620	227.1431	0	0	0.0518615	5.931510	NA	NA	NA	NA	- 0.3836941
preserved ~ CumPres	1447.351	250.8748	0	0	0.0376986	2.283010	0.3305372	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1484.607	288.1302	0	0	0.0058200	2.673665	NA	NA	- 0.0275324	0.1858274	NA
preserved ~ pos	1485.719	289.2429	0	0	0.0025685	3.063332	NA	NA	NA	- 0.0582450	NA
preserved ~ 1	1487.567	291.0907	0	0	0.0000000	2.833527	NA	NA	NA	NA	NA

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

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

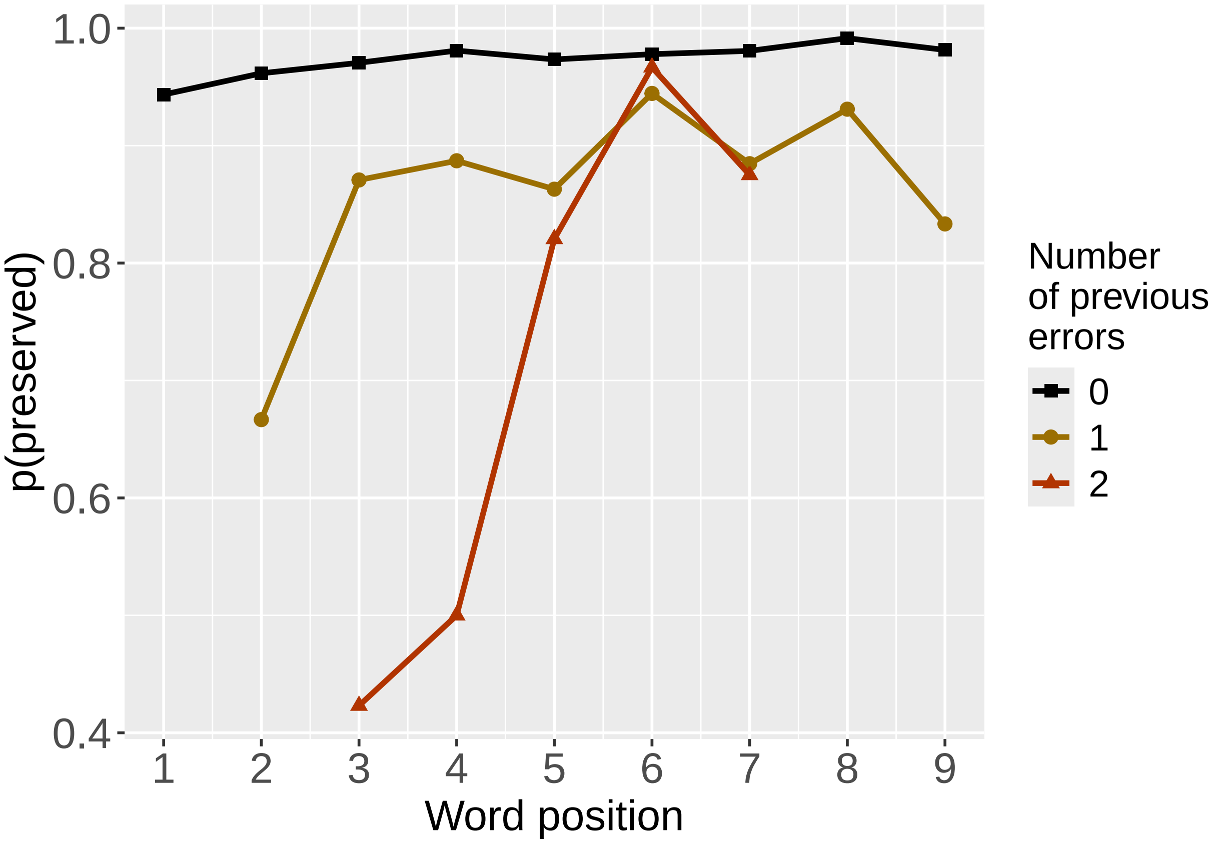
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

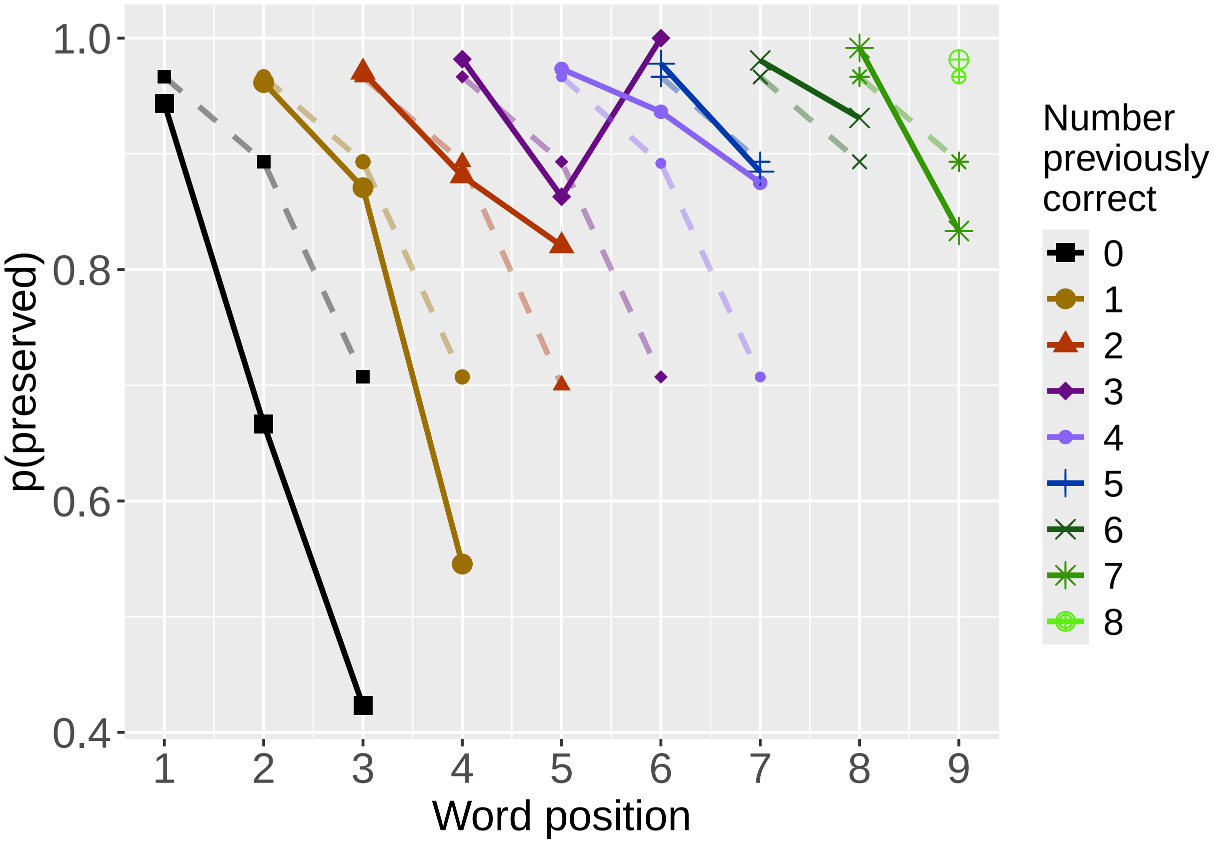
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

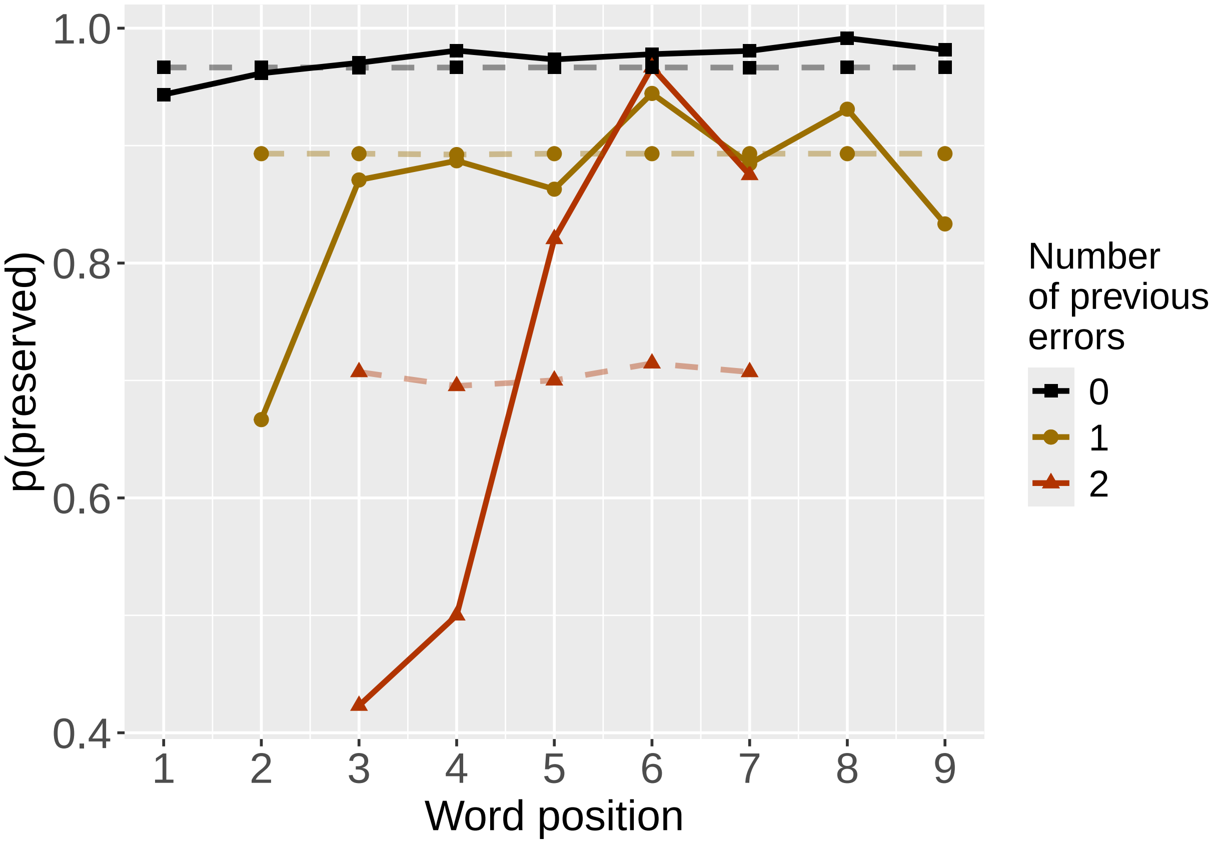
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.44612     -1.46143     -0.01326      0.33845
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1416  AIC: 1487
## log likelihood:  -708.004
## Nagelkerke R2:  0.2563749
## % pres/err predicted correctly:  -349.1464
## % of predictable range [ (model-null)/(1-null) ]:  0.2247509
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.365       -1.241
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1449  AIC: 1517
## log likelihood:  -724.6516
## Nagelkerke R2:  0.2357118
## % pres/err predicted correctly:  -355.6997
## % of predictable range [ (model-null)/(1-null) ]:  0.2102416
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.62288     -0.02958      0.20082
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1802  AIC: 1895
## log likelihood:  -901.0595
## Nagelkerke R2:  0.00644016
## % pres/err predicted correctly:  -449.4897
## % of predictable range [ (model-null)/(1-null) ]:  0.002583831
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1486.638	0.00000	1e+00	0.9999997	0.2563749	2.446116	-1.461434	-0.0132602	0.3384474

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	1516.649	30.01123	3e-07	0.0000003	0.2357118	3.364632	-1.241392	NA	NA
preserved ~ I(pos^2) + pos	1895.114	408.47592	0e+00	0.0000000	0.0064402	2.622876	NA	-0.0295770	0.2008221

```r
########
# 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
##      4.9942      -1.1555      -0.2077
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1432  AIC: 1498
## log likelihood:  -715.9135
## Nagelkerke R2:  0.2465779
## % pres/err predicted correctly:  -354.2298
## % of predictable range [ (model-null)/(1-null) ]:  0.2134959
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.365       -1.241
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1449  AIC: 1517
## log likelihood:  -724.6516
## Nagelkerke R2:  0.2357118
## % pres/err predicted correctly:  -355.6997
## % of predictable range [ (model-null)/(1-null) ]:  0.2102416
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       5.879       -0.380
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1736  AIC: 1819
## log likelihood:  -867.8829
## Nagelkerke R2:  0.05103445
## % pres/err predicted correctly:  -442.1959
## % of predictable range [ (model-null)/(1-null) ]:  0.01873292
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	1498.148	0.00000	1.0e+00	0.999904	0.2465779	4.994225	-1.155524	-0.2077231
preserved ~ CumErr	1516.649	18.50168	9.6e-05	0.000096	0.2357118	3.364632	-1.241392	NA
preserved ~ stimlen	1818.518	320.37072	0.0e+00	0.000000	0.0510345	5.878743	NA	-0.3799683

```
########
# 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
##      2.8361      -1.2292       0.2298
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1417  AIC: 1485
## log likelihood:  -708.3173
## Nagelkerke R2:  0.2559875
## % pres/err predicted correctly:  -348.9125
## % of predictable range [ (model-null)/(1-null) ]:  0.2252689
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.365       -1.241
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1449  AIC: 1517
## log likelihood:  -724.6516
## Nagelkerke R2:  0.2357118
## % pres/err predicted correctly:  -355.6997
## % of predictable range [ (model-null)/(1-null) ]:  0.2102416
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.2023       0.2828
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1751  AIC: 1841
## log likelihood:  -875.4459
## Nagelkerke R2:  0.0409303
## % pres/err predicted correctly:  -443.8285
## % of predictable range [ (model-null)/(1-null) ]:  0.01511815
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	1485.170	0.00000	1e+00	0.9999999	0.2559875	2.836110	-1.229171	0.2298328
preserved ~ CumErr	1516.649	31.47894	1e-07	0.0000001	0.2357118	3.364632	-1.241392	NA
preserved ~ CumPres	1841.236	356.06527	0e+00	0.0000000	0.0409303	2.202342	NA	0.2827530

```
########
# 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
##      2.6063      -1.4590       0.2298
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1417  AIC: 1485
## log likelihood:  -708.3173
## Nagelkerke R2:  0.2559875
## % pres/err predicted correctly:  -348.9125
## % of predictable range [ (model-null)/(1-null) ]:  0.2252689
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.365       -1.241
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1449  AIC: 1517
## log likelihood:  -724.6516
## Nagelkerke R2:  0.2357118
## % pres/err predicted correctly:  -355.6997
## % of predictable range [ (model-null)/(1-null) ]:  0.2102416
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     3.05823     -0.06339
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1807  AIC: 1898
## log likelihood:  -903.6752
## Nagelkerke R2:  0.002894437
## % pres/err predicted correctly:  -450.1118
## % of predictable range [ (model-null)/(1-null) ]:  0.001206399
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	1485.170	0.00000	1e+00	0.9999999	0.2559875	2.606277	-1.459003	0.2298328
preserved ~ CumErr	1516.649	31.47894	1e-07	0.0000001	0.2357118	3.364632	-1.241392	NA
preserved ~ pos	1897.597	412.42654	0e+00	0.0000000	0.0028944	3.058232	NA	-0.0633864

```
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	1485.170	0.00000	1.0e+00	0.9999999	0.2559875	2.606277	-1.459003	NA	0.2298328	NA	NA
preserved ~ CumErr + CumPres	1485.170	0.00000	1.0e+00	0.9999999	0.2559875	2.836110	-1.229171	NA	NA	NA	0.2298328
preserved ~ CumErr + I(pos^2) + pos	1486.638	0.00000	1.0e+00	0.9999997	0.2563742	2.446116	-1.461434	-0.0132602	0.3384474	NA	NA
preserved ~ CumErr + stimlen	1498.148	0.00000	1.0e+00	0.9999040	0.2465774	4.994225	-1.155524	NA	NA	-0.2077231	NA
preserved ~ CumErr	1516.649	30.01123	3.0e-07	0.0000003	0.2357118	3.364632	-1.241392	NA	NA	NA	NA
preserved ~ CumErr	1516.649	18.50168	9.6e-05	0.0000960	0.2357118	3.364632	-1.241392	NA	NA	NA	NA
preserved ~ CumErr	1516.649	31.47894	1.0e-07	0.0000001	0.2357118	3.364632	-1.241392	NA	NA	NA	NA
preserved ~ CumErr	1516.649	31.47894	1.0e-07	0.0000001	0.2357118	3.364632	-1.241392	NA	NA	NA	NA
preserved ~ stimlen	1818.518	320.37072	0.0e+00	0.0000000	0.0510345	5.878743	NA	NA	NA	-0.3799683	NA
preserved ~ CumPres	1841.236	356.06527	0.0e+00	0.0000000	0.0409302	3.202342	NA	NA	NA	NA	0.2827530
preserved ~ I(pos^2) + pos	1895.114	408.47592	0.0e+00	0.0000000	0.0064402	2.622876	NA	-0.0295770	0.2008221	NA	NA
preserved ~ pos	1897.597	412.42654	0.0e+00	0.0000000	0.0028944	3.058232	NA	NA	-0.0633864	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
##      4.5346      -1.3593       0.2901      -0.2656       0.1555
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4216 Residual
## Null Deviance:       1812
## Residual Deviance: 1365  AIC: 1434
## log likelihood:  -682.6333
## Nagelkerke R2:  0.2875534
## % pres/err predicted correctly:  -343.3745
## % of predictable range [ (model-null)/(1-null) ]:  0.2375304
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen
##      4.8086      -1.3867       0.2873      -0.3063
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1380  AIC: 1445
## log likelihood:  -689.7931
## Nagelkerke R2:  0.2787925
## % pres/err predicted correctly:  -343.5789
## % of predictable range [ (model-null)/(1-null) ]:  0.2370779
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq
##      2.6138      -1.4076       0.2470       0.2026
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1391  AIC: 1462
## log likelihood:  -695.4855
## Nagelkerke R2:  0.2718058
## % pres/err predicted correctly:  -347.8865
## % of predictable range [ (model-null)/(1-null) ]:  0.2275406
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.6063      -1.4590       0.2298
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1417  AIC: 1485
## log likelihood:  -708.3173
## Nagelkerke R2:  0.2559875
## % pres/err predicted correctly:  -348.9125
## % of predictable range [ (model-null)/(1-null) ]:  0.2252689
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       2.808
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4220 Residual
## Null Deviance:       1812
## Residual Deviance: 1812  AIC: 1900
## log likelihood:  -905.808
## Nagelkerke R2:  6.362956e-16
## % pres/err predicted correctly:  -450.6567
## % 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	1433.594	0.00000	1.0000000	0.9970492	0.2875534	1.534641	-1.359299	0.2900596	0.1554858	-0.2655837
preserved ~ CumErr + pos + stimlen	1445.240	11.64602	0.0029587	0.0029500	0.2787924	1.808552	-1.386708	0.2873224	NA	-0.3063300
preserved ~ CumErr + pos + log_freq	1461.611	28.01672	0.0000008	0.0000008	0.2718052	2.613816	-1.407572	0.2470397	0.2025586	NA
preserved ~ CumErr + pos	1485.170	51.57643	0.0000000	0.0000000	0.2559872	2.606277	-1.459003	0.2298328	NA	NA
preserved ~ 1	1900.497	466.9025	0.0000000	0.0000000	0.0000000	2.807913	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!
```

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

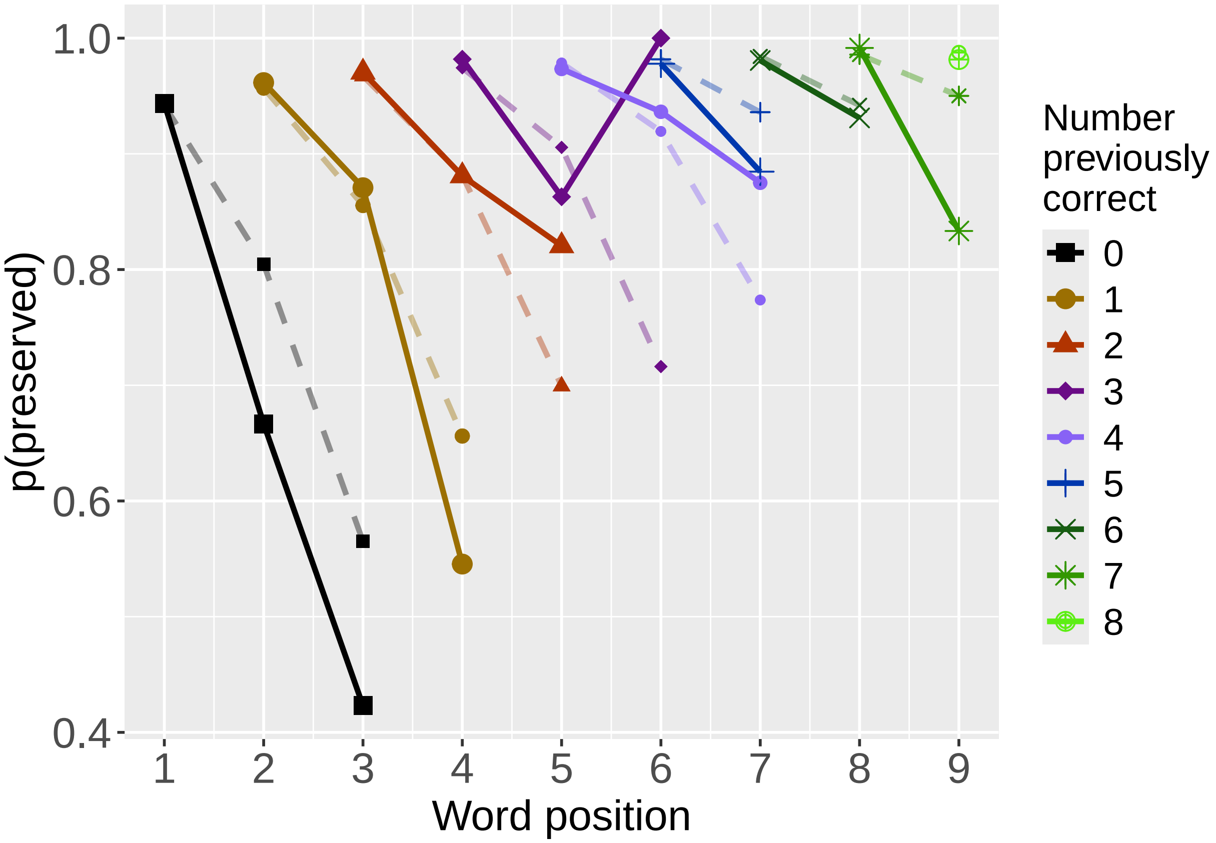
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   1707.5 1773.8
## pos       1   1418.0 1484.4
## stimlen   1   1391.0 1457.3
## log_freq  1   1379.6 1445.9
## <none>        1365.3 1433.6
```

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

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

```r
# 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.
```

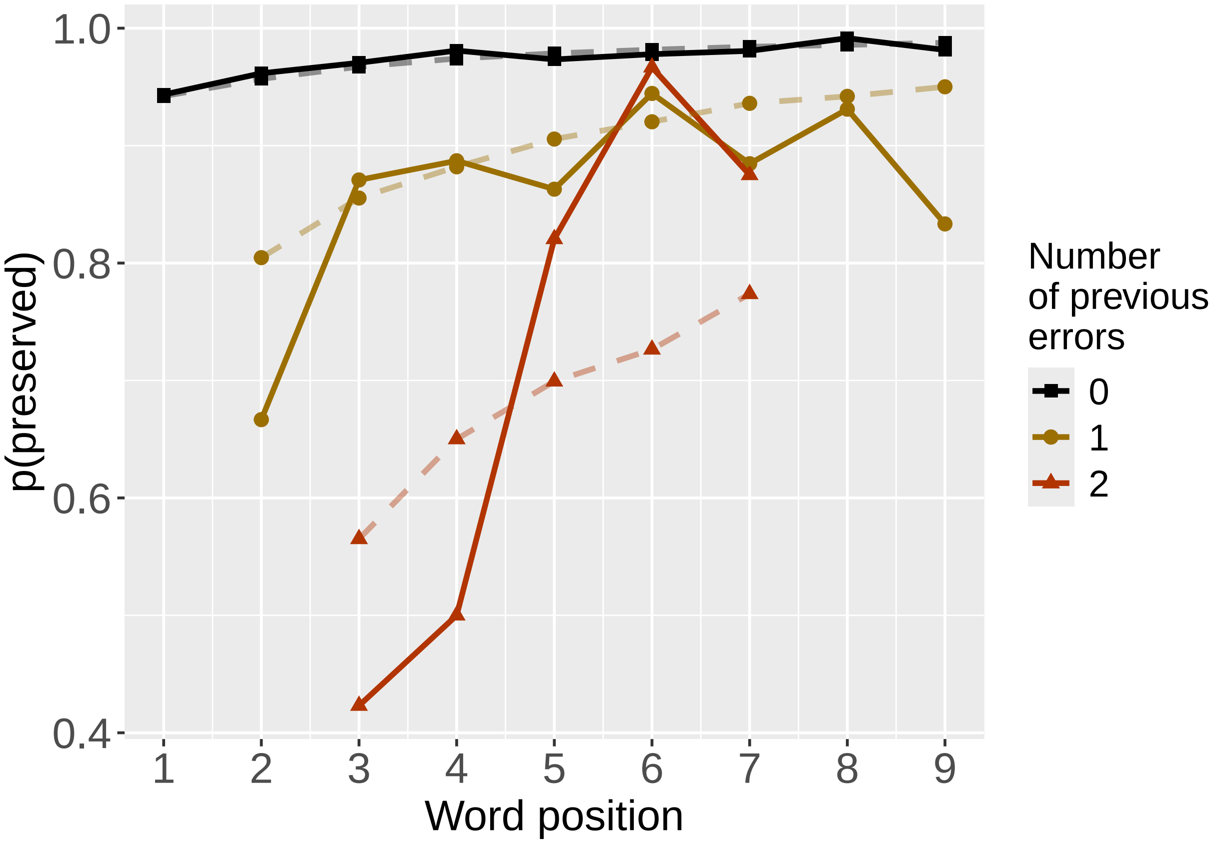
```r
print(PrevCorPlot)
```



```
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
##       3.365       -1.241
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4219 Residual
## Null Deviance:       1812
## Residual Deviance: 1449  AIC: 1517
## log likelihood:  -724.6516
## Nagelkerke R2:  0.2357118
## % pres/err predicted correctly:  -355.6997
## % of predictable range [ (model-null)/(1-null) ]:  0.2102416
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.6063      -1.4590       0.2298
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4218 Residual
## Null Deviance:       1812
## Residual Deviance: 1417  AIC: 1485
## log likelihood:  -708.3173
## Nagelkerke R2:  0.2559875
## % pres/err predicted correctly:  -348.9125
## % of predictable range [ (model-null)/(1-null) ]:  0.2252689
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen
##      4.8086      -1.3867       0.2873      -0.3063
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4217 Residual
## Null Deviance:       1812
## Residual Deviance: 1380  AIC: 1445
## log likelihood:  -689.7931
## Nagelkerke R2:  0.2787925
## % pres/err predicted correctly:  -343.5789
## % of predictable range [ (model-null)/(1-null) ]:  0.2370779
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen     log_freq
##      4.5346      -1.3593       0.2901      -0.2656       0.1555
##
## Degrees of Freedom: 4220 Total (i.e. Null);  4216 Residual
## Null Deviance:       1812
## Residual Deviance: 1365  AIC: 1434
## log likelihood:  -682.6333
## Nagelkerke R2:  0.2875534
## % pres/err predicted correctly:  -343.3745
## % of predictable range [ (model-null)/(1-null) ]:  0.2375304
## ***************************
```

```
## `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 5 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 rows containing missing values or values outside the scale range (`geom_point()`).
```

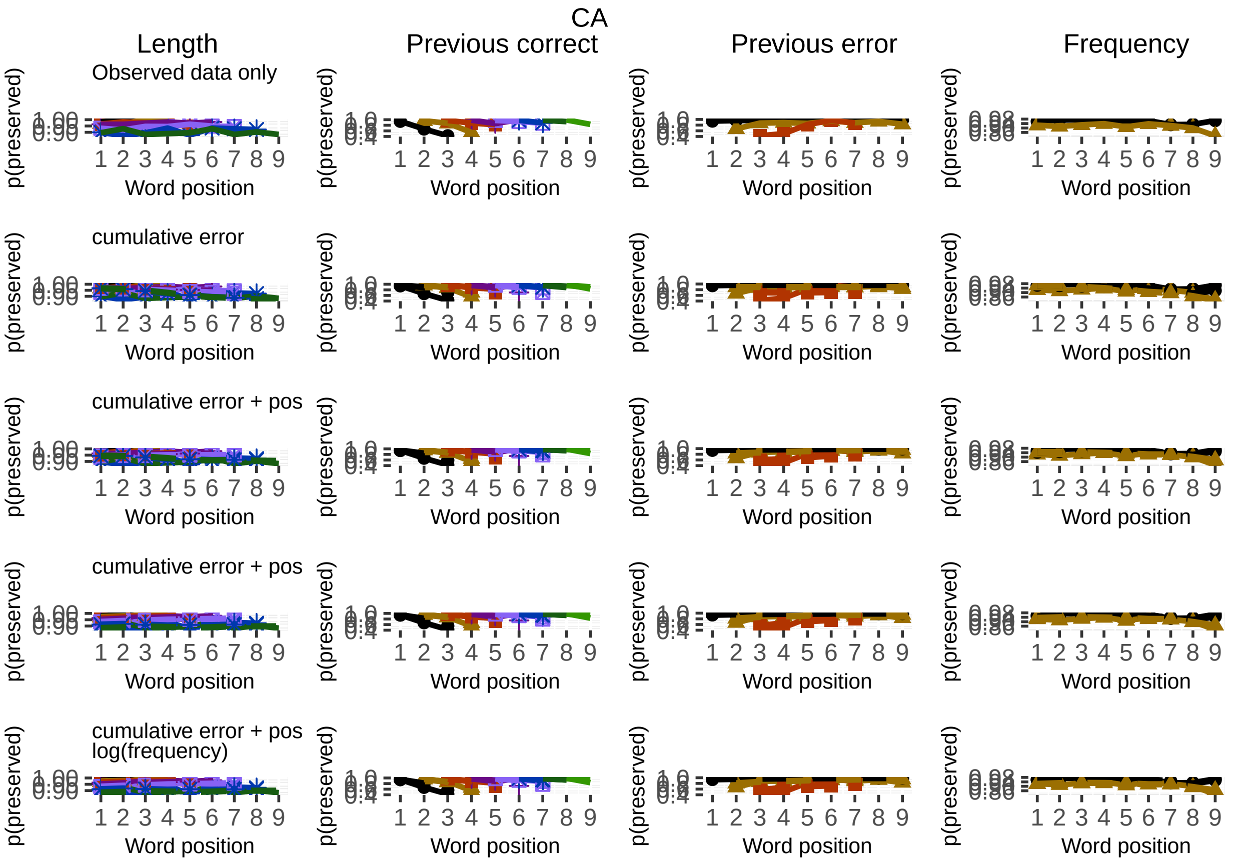
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 rows containing missing values or values outside the scale range (`geom_point()`).
```

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



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

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

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

	CumErr	pos	stimlen	log_freq
McFadden	0.1934217	0.0137292	0.0267791	0.0162166
SquaredCorrelation	0.0821509	0.0056042	0.0116093	0.0070468
Nagelkerke	0.0821509	0.0056042	0.0116093	0.0070468
Estrella	0.0940083	0.0070736	0.0126989	0.0076670

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

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

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

```
##                                                               model deviance
## CumErr + pos + stimlen + log_freq CumErr + pos + stimlen + log_freq 1365.267
## CumErr + pos + stimlen                       CumErr + pos + stimlen 1379.586
## CumErr + pos                                           CumErr + pos 1416.635
## CumErr                                                       CumErr 1449.303
## null                                                           null 1811.616
##                                   deviance_explained percent_explained
## CumErr + pos + stimlen + log_freq           446.3495          24.63819
## CumErr + pos + stimlen                      432.0298          23.84776
## CumErr + pos                                394.9814          21.80271
## CumErr                                      362.3128          19.99943
## null                                          0.0000           0.00000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + pos + stimlen + log_freq                     100.00000               3.208171
## CumErr + pos + stimlen                                 96.79183               8.300321
## CumErr + pos                                           88.49151               7.319053
## CumErr                                                 81.17246              81.172456
## null                                                         NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + stimlen + log_freq	1365.267	446.3495
CumErr + pos + stimlen	1379.586	432.0298
CumErr + pos	1416.635	394.9814
CumErr	1449.303	362.3128
null	1811.616	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + stimlen + log_freq	24.63819	100.00000	3.208171
CumErr + pos + stimlen	23.84776	96.79183	8.300320
CumErr + pos	21.80271	88.49151	7.319053
CumErr	19.99943	81.17246	81.172456
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   0.77201402
## pos      0.05266521
## stimlen  0.10909873
## log_freq 0.06622204
```

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

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

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

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.7180496	1449.303
preserved ~ CumErr+pos	0.8511361	1416.635
preserved ~ CumErr+pos+stimlen+log_freq	0.8899984	1365.267
preserved ~ CumErr+pos+stimlen	0.8914402	1379.586

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

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

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

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

```
##                                       model p_accounted_for model_deviance diff_CumErr
## 1                        preserved ~ CumErr       0.7180496       1449.303   0.0000000
## 2                    preserved ~ CumErr+pos       0.8511361       1416.635   0.1330865
## 3 preserved ~ CumErr+pos+stimlen+log_freq       0.8899984       1365.267   0.1719488
## 4            preserved ~ CumErr+pos+stimlen       0.8914402       1379.586   0.1733906
##   diff_CumErr+pos diff_CumErr+pos+stimlen+log_freq diff_CumErr+pos+stimlen
## 1     -0.13308653                     -0.171948807            -0.173390641
## 2      0.00000000                     -0.038862281            -0.040304115
## 3      0.03886228                      0.000000000            -0.001441834
## 4      0.04030412                      0.001441834             0.000000000
```

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

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

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

model	diff_CumErr	diff_CumErr+pos	diff_CumErr+pos+stimlen+log_freq
preserved ~ CumErr	0.0000000	-0.1330865	-0.1719488
preserved ~ CumErr+pos	0.1330865	0.0000000	-0.0388623
preserved ~ CumErr+pos+stimlen+log_freq	0.1719488	0.0388623	0.0000000
preserved ~ CumErr+pos+stimlen	0.1733906	0.0403041	0.0014418