GC - 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	548	35	132	NA	NA	715
2	66	NA	438	101	110	715
3	318	NA	171	210	16	715
4	304	NA	246	69	38	657
5	238	NA	214	74	39	565
6	210	1	139	73	23	446
7	179	NA	105	29	19	332
8	93	NA	56	26	4	179
9	77	NA	2	NA	7	86

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7664336	0.0489510	0.1846154	NA	NA	715
2	0.0923077	NA	0.6125874	0.1412587	0.1538462	715
3	0.4447552	NA	0.2391608	0.2937063	0.0223776	715
4	0.4627093	NA	0.3744292	0.1050228	0.0578387	657
5	0.4212389	NA	0.3787611	0.1309735	0.0690265	565
6	0.4708520	0.0022422	0.3116592	0.1636771	0.0515695	446

pos_factor	O	P	V	1	S	total
7	0.5391566	NA	0.3162651	0.0873494	0.0572289	332
8	0.5195531	NA	0.3128492	0.1452514	0.0223464	179
9	0.8953488	NA	0.0232558	NA	0.0813953	86

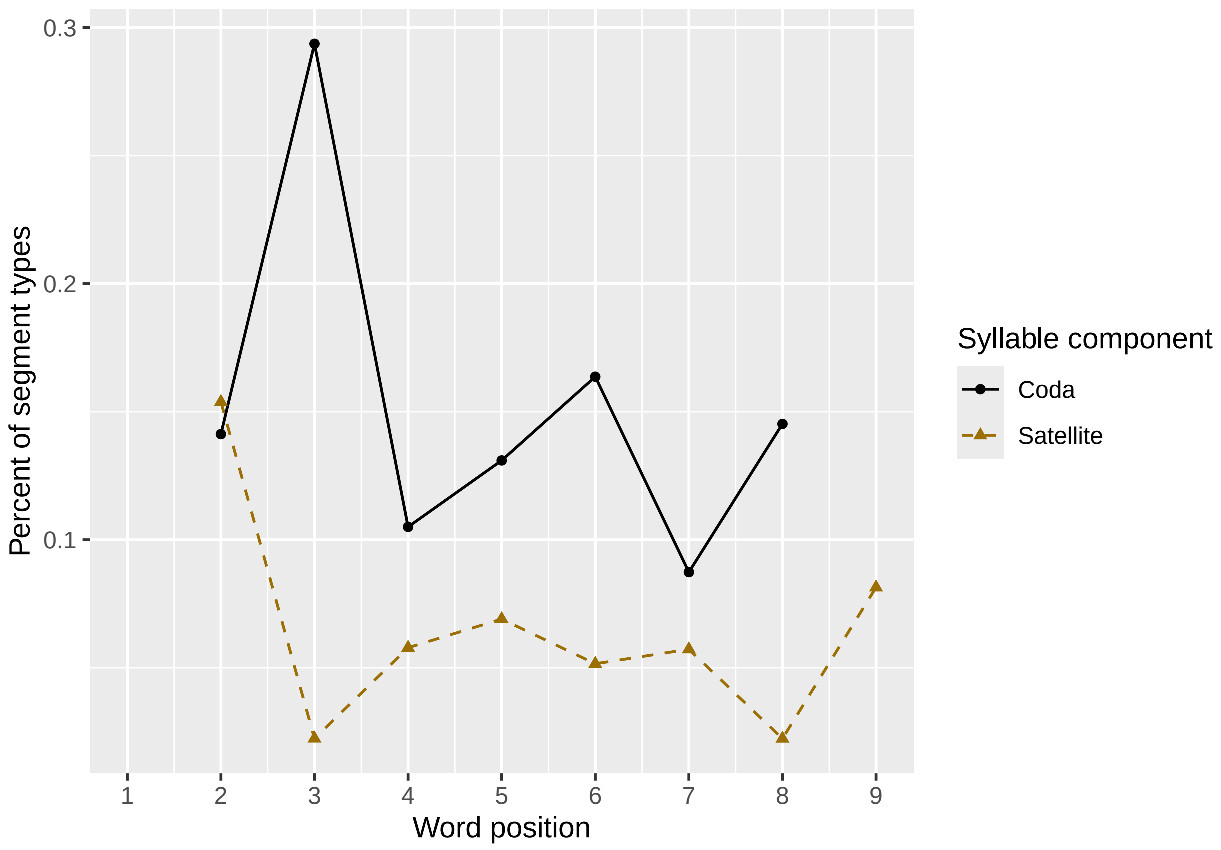
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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()`).
```



```r
# 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.
```

```r
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 0.966 1     0.948 NA     NA     NA     NA     NA     NA
## 2       5 0.826 0.962 0.880  0.978 NA     NA     NA     NA     NA
## 3       6 0.849 0.924 0.950  0.929  0.954 NA     NA     NA     NA
## 4       7 0.833 0.930 0.895  0.921  0.947  0.965 NA     NA     NA
## 5       8 0.856 0.941 0.903  0.881  0.937  0.915  0.959 NA     NA
## 6       9 0.892 0.968 0.925  0.930  0.903  0.941  0.946  0.968 NA
## 7      10 0.895 1     0.884  0.849  0.942  0.919  0.953  0.930  0.988
```

```r
# 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.
```

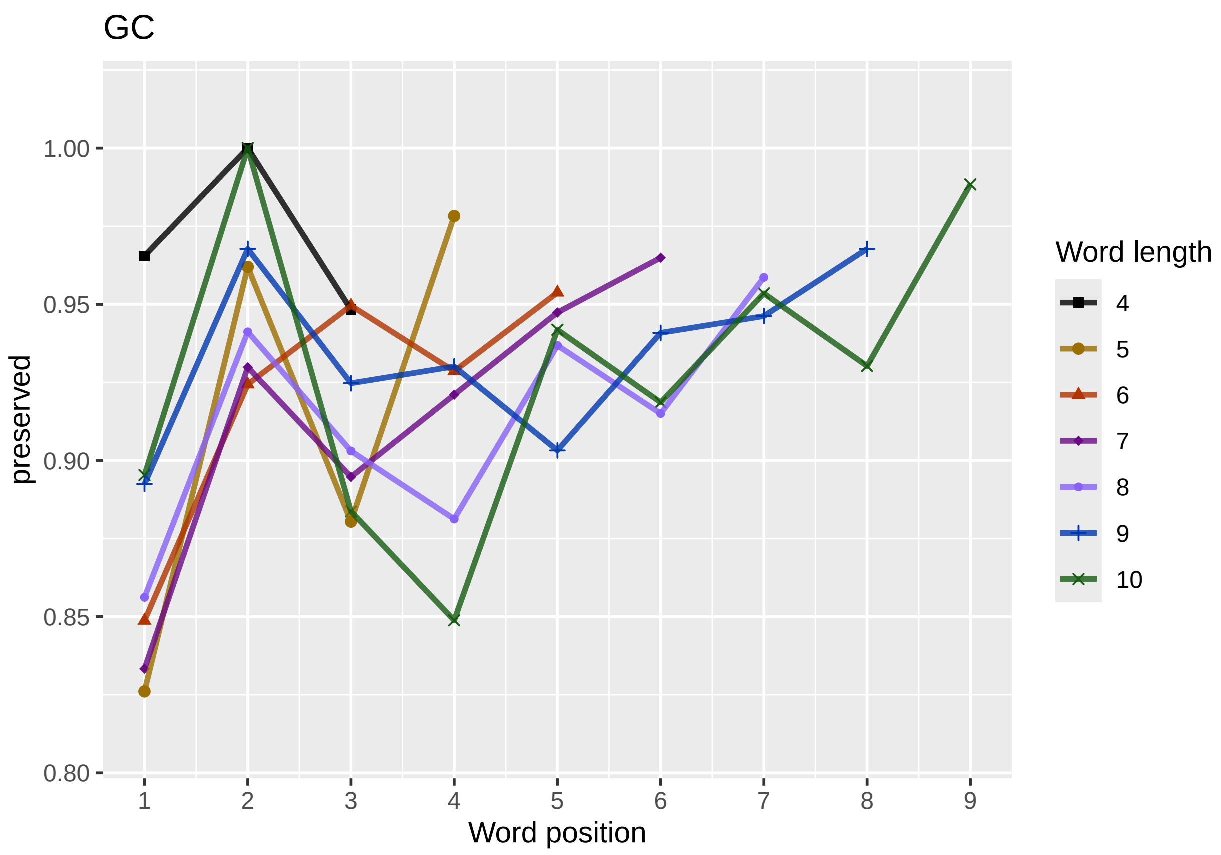
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    92    92    92    92    NA    NA    NA    NA    NA
## 3       6   119   119   119   119   119    NA    NA    NA    NA
## 4       7   114   114   114   114   114   114    NA    NA    NA
## 5       8   153   153   153   153   153   153   153    NA    NA
## 6       9    93    93    93    93    93    93    93    93    NA
## 7      10    86    86    86    86    86    86    86    86    86
```

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

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

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



Length and position

```
# length and position

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

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

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

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

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         0.24288           0.22171          -0.15618           1.52618           0.01957
##     stimlen:pos
##        -0.17545
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4404 Residual
## Null Deviance:       2370
## Residual Deviance: 2335  AIC: 2403
## log likelihood:  -1167.643
## Nagelkerke R2:  0.01892059
## % pres/err predicted correctly:  -619.5869
## % of predictable range [ (model-null)/(1-null) ]:  0.007726202
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.4076      -0.0611       0.1519
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2343  AIC: 2404
## log likelihood:  -1171.486
## Nagelkerke R2:  0.01475779
## % pres/err predicted correctly:  -620.6666
## % of predictable range [ (model-null)/(1-null) ]:  0.005999847
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.76303      0.01747      0.38968     -0.02810
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2341  AIC: 2404
## log likelihood:  -1170.388
## Nagelkerke R2:  0.01594777
## % pres/err predicted correctly:  -620.2758
## % of predictable range [ (model-null)/(1-null) ]:  0.006624768
## ***************************
## model index:  3
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.9979       0.1354
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2405
## log likelihood:  -1173
## Nagelkerke R2:  0.01311572
## % pres/err predicted correctly:  -621.03
## % of predictable range [ (model-null)/(1-null) ]:  0.005418915
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.423565    -0.061542     0.001101     0.143092
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2343  AIC: 2406
## log likelihood:  -1171.483
## Nagelkerke R2:  0.01476171
## % pres/err predicted correctly:  -620.6775
## % of predictable range [ (model-null)/(1-null) ]:  0.005982464
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.971113    -0.002281     0.153813
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2407
## log likelihood:  -1172.985
## Nagelkerke R2:  0.01313287
## % pres/err predicted correctly:  -620.9994
## % of predictable range [ (model-null)/(1-null) ]:  0.005467733
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##       2.484
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4409 Residual
## Null Deviance:      2370
## Residual Deviance: 2370  AIC: 2428
## log likelihood:  -1185.057
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -624.419
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    2.448753     0.004595
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:      2370
## Residual Deviance: 2370  AIC: 2429
## log likelihood:  -1185.048
## Nagelkerke R2:  1.029884e-05
## % pres/err predicted correctly:  -624.4153
## % of predictable range [ (model-null)/(1-null) ]:  5.940735e-06
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	2402.620	0.000000	1.0000000	0.3644854	0.0189206	2.428780	0.2217139	0.5261775	-0.1754480	-0.1561752	0.0195737
preserved ~ stimlen + pos	2403.851	1.230500	0.5405057	0.1970064	0.0147572	2.4075986	-0.0610988	0.1518593	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	2403.957	1.330165	0.5142301	0.1874297	0.0159471	1.8763033	0.0174658	0.3896802	-0.0281045	NA	NA
preserved ~ pos	2404.686	2.058494	0.3572759	0.1302218	0.0131115	1.7997935	NA	0.1354309	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	2405.813	3.185576	0.2033570	0.0741210	0.0147612	2.4235654	-0.0615424	0.1430925	NA	0.0011008	NA
preserved ~ I(pos^2) + pos	2406.734	4.108041	0.1282184	0.0467330	0.0131329	1.9711135	NA	0.1538133	NA	-0.0022809	NA
preserved ~ 1	2427.522	24.893553	0.0000039	0.0000014	0.0000000	2.4840472	NA	NA	NA	NA	NA
preserved ~ stimlen	2429.479	26.852366	0.0000015	0.0000005	0.0000102	2.4487534	-0.0045949	NA	NA	NA	NA

```r
print(BestLPModelFormula)
```

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

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)            stimlen           I(pos^2)                pos  stimlen:I(pos^2)
##          0.24288            0.22171           -0.15618            1.52618           0.01957
##      stimlen:pos
##         -0.17545
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4404 Residual
## Null Deviance:       2370
## Residual Deviance: 2335  AIC: 2403
```

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

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

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

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

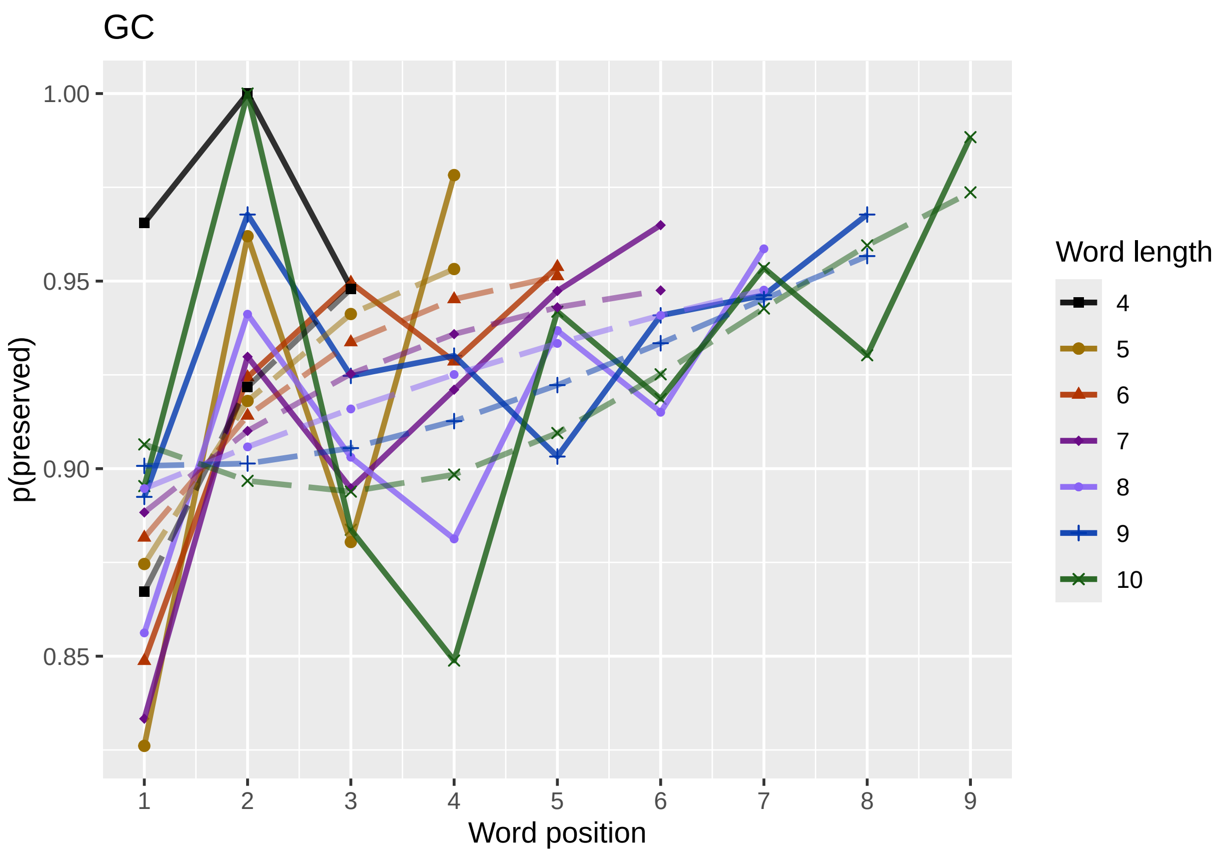
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.867 0.922 0.948 NA     NA     NA     NA     NA     NA
## 2       5 0.875 0.918 0.941  0.953 NA     NA     NA     NA     NA
## 3       6 0.882 0.914 0.934  0.945  0.951 NA     NA     NA     NA
## 4       7 0.888 0.910 0.925  0.936  0.943  0.948 NA     NA     NA
## 5        8 0.895 0.906 0.916  0.925  0.933  0.941  0.948 NA     NA
## 6        9 0.901 0.901 0.905  0.913  0.922  0.933  0.945  0.957 NA
## 7       10 0.906 0.897 0.894  0.898  0.909  0.925  0.943  0.960  0.974
```

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

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

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

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



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

```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        7   715
```

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

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

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

```
## [1] "The number of responses with fragments was 7 / 715 = 0.98 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:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##     (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         0.45759           0.20258          -0.11517           1.30646           0.01639
##     stimlen:pos
##        -0.15665
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4387 Residual
## Null Deviance:       2282
## Residual Deviance: 2234  AIC: 2301
## log likelihood:  -1116.88
## Nagelkerke R2:  0.02702722
## % pres/err predicted correctly:  -589.2465
## % of predictable range [ (model-null)/(1-null) ]:  0.01031947
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.27735     -0.05505      0.19383
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4390 Residual
## Null Deviance:       2282
## Residual Deviance: 2241  AIC: 2302
## log likelihood:  -1120.467
## Nagelkerke R2:  0.02303758
## % pres/err predicted correctly:  -590.0311
## % of predictable range [ (model-null)/(1-null) ]:  0.009003975
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.9069       0.1796
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4391 Residual
## Null Deviance:       2282
## Residual Deviance: 2243  AIC: 2303
## log likelihood:  -1121.666
## Nagelkerke R2:  0.02170304
## % pres/err predicted correctly:  -590.3003
## % of predictable range [ (model-null)/(1-null) ]:  0.008552629
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.73147      0.01185      0.40192     -0.02476
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4389 Residual
## Null Deviance:       2282
## Residual Deviance: 2239  AIC: 2303
## log likelihood:  -1119.695
## Nagelkerke R2:  0.02389663
## % pres/err predicted correctly:  -589.7207
## % of predictable range [ (model-null)/(1-null) ]:  0.009524382
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.42711     -0.05903      0.01108      0.10814
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4389 Residual
## Null Deviance:       2282
## Residual Deviance: 2240  AIC: 2303
## log likelihood:  -1120.158
## Nagelkerke R2:  0.0233815
## % pres/err predicted correctly:  -590.1208
## % of predictable range [ (model-null)/(1-null) ]:  0.008853635
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.992501     0.007743     0.119094
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4390 Residual
## Null Deviance:       2282
## Residual Deviance: 2243  AIC: 2304
## log likelihood:  -1121.512
## Nagelkerke R2:  0.02187396
## % pres/err predicted correctly:  -590.3829
## % of predictable range [ (model-null)/(1-null) ]:  0.00841417
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.535
```

```
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4392 Residual
## Null Deviance:       2282
## Residual Deviance: 2282  AIC: 2340
## log likelihood:  -1141.066
## Nagelkerke R2:  -5.480175e-16
## % pres/err predicted correctly:  -595.4011
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.34721      0.02459
##
## Degrees of Freedom: 4392 Total (i.e. Null);  4391 Residual
## Null Deviance:       2282
## Residual Deviance: 2282  AIC: 2342
## log likelihood:  -1140.806
## Nagelkerke R2:  0.0002919691
## % pres/err predicted correctly:  -595.3288
## % of predictable range [ (model-null)/(1-null) ]:  0.0001212272
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	2301.219	0.000000	1.0000000	0.3358938	0.0270272	2.4575876	0.2025755	1.3064576	-0.1566520	-0.1151721	0.0163888
preserved ~ stimlen + pos	2302.436	1.217072	0.5441470	0.1827756	0.0230376	2.2773502	-0.0550467	0.1938314	NA	NA	NA
preserved ~ pos	2302.658	1.439061	0.4869809	0.1635739	0.0217031	2.9068981	NA	0.1795883	NA	NA	NA
preserved ~ stimlen * pos	2303.159	1.939754	0.3791297	0.1273470	0.0238966	1.6731466	0.0118481	1.4019217	-0.0247603	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	2303.442	2.228775	0.3281101	0.1102122	0.0233812	2.4271106	-0.0590288	0.1081446	NA	0.0110793	NA
preserved ~ I(pos^2) + pos	2304.082	2.864612	0.2387577	0.0801972	0.0218740	1.9925009	NA	0.1190936	NA	0.0077427	NA
preserved ~ 1	2340.207	38.987935	0.0000000	0.0000000	0.0000000	2.5354180	NA	NA	NA	NA	NA
preserved ~ stimlen	2341.574	40.357993	0.0000000	0.0000000	0.0002920	2.3472093	0.0245946	NA	NA	NA	NA

```r
# plot no fragment data

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

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

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

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

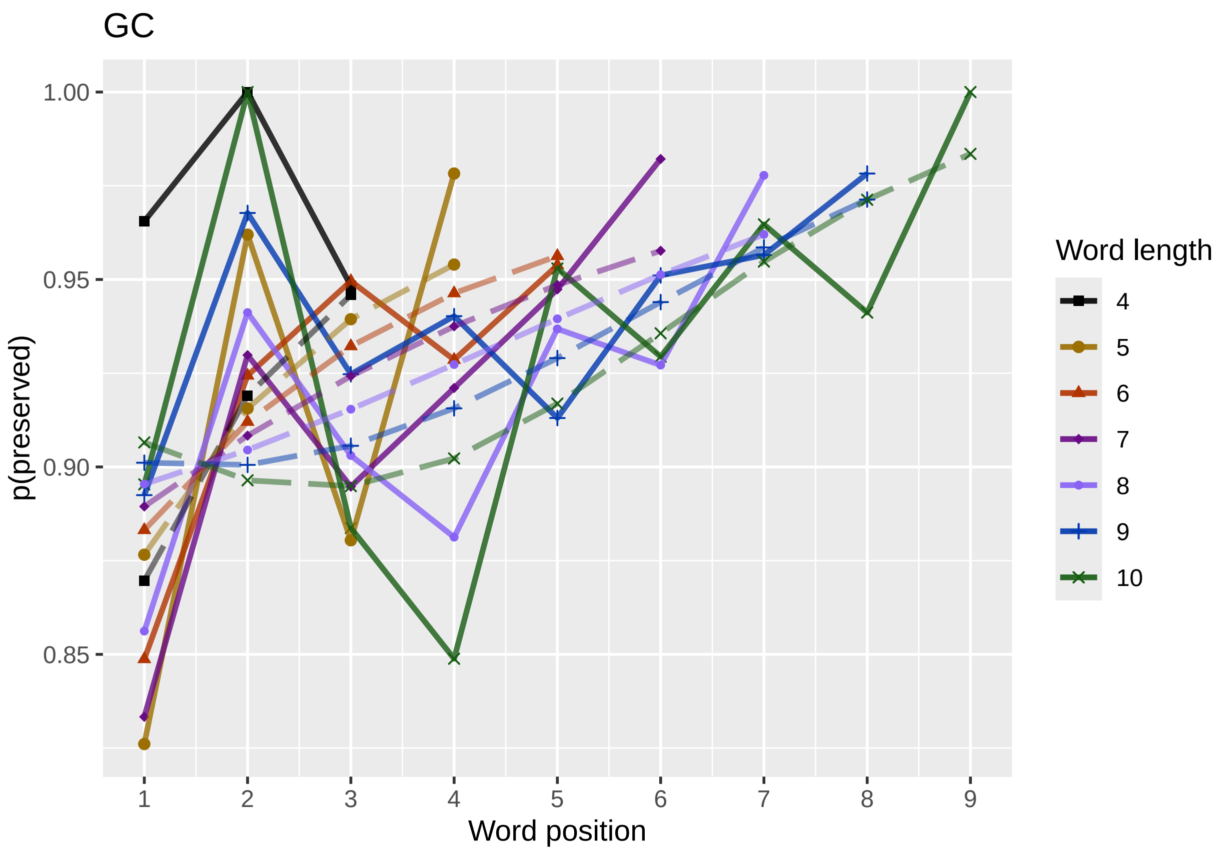
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.870 0.919 0.946 NA     NA     NA     NA     NA     NA
## 2       5 0.877 0.916 0.939  0.954 NA     NA     NA     NA     NA
## 3       6 0.883 0.912 0.932  0.946  0.956 NA     NA     NA     NA
## 4       7 0.889 0.908 0.924  0.937  0.949  0.958 NA     NA     NA
## 5       8 0.895 0.905 0.915  0.927  0.940  0.951  0.962 NA     NA
## 6       9 0.901 0.901 0.906  0.916  0.929  0.944  0.959  0.971 NA
## 7      10 0.907 0.896 0.895  0.902  0.917  0.936  0.955  0.971  0.984
```

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

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

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

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"no_fragments_percent_preserved_by_length_pos_wfit.png"),plot=no
nofrag_fitted_len_pos_plot
```



back to full data

```
results_report_DF <- AddReportLine(results_report_DF,"min preserved",min_preserved)
results_report_DF <- AddReportLine(results_report_DF,"max preserved",max_preserved)
print(sprintf("Min/max preserved range: %.2f - %.2f",min_preserved,max_preserved))
```

```
## [1] "Min/max preserved range: 0.81 - 1.02"
```

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

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

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

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

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.004372215
```

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

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

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

```r
print(OA_mean_pos_diff)
```

```
## [1] 0.01391762
```

```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"
```

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

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

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

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

  print(" ")
  # percentage return upward
  percentage_return_upward <- upward_dist[biggest_return_upward_row]/downward_dist[biggest_return_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:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.01550       0.13505       0.22472      -0.02586
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2321  AIC: 2388
## log likelihood:  -1160.722
## Nagelkerke R2:  0.02639888
## % pres/err predicted correctly:  -616.7838
## % of predictable range [ (model-null)/(1-null) ]:  0.01220815
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      1.9773       0.1483       0.1389
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2324  AIC: 2389
## log likelihood:  -1162.188
## Nagelkerke R2:  0.02481721
## % pres/err predicted correctly:  -617.547
## % of predictable range [ (model-null)/(1-null) ]:  0.01098796
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.10288          -0.02182           0.38508           0.15268          -0.03256
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4405 Residual
## Null Deviance:       2370
## Residual Deviance: 2321  AIC: 2389
## log likelihood:  -1160.447
## Nagelkerke R2:  0.02669551
## % pres/err predicted correctly:  -617.0692
## % of predictable range [ (model-null)/(1-null) ]:  0.01175189
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.10093      -0.01276       0.13841       0.22019      -0.02538
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4405 Residual
## Null Deviance:       2370
## Residual Deviance: 2321  AIC: 2389
## log likelihood:  -1160.663
## Nagelkerke R2:  0.02646314
## % pres/err predicted correctly:  -616.8062
## % of predictable range [ (model-null)/(1-null) ]:  0.01217242
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.10482          -0.01763           0.39635           0.14288          -0.02574
##     log_freq:pos
##         -0.01894
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4404 Residual
## Null Deviance:       2370
## Residual Deviance: 2319  AIC: 2390
## log likelihood:  -1159.745
## Nagelkerke R2:  0.02745233
## % pres/err predicted correctly:  -616.599
## % of predictable range [ (model-null)/(1-null) ]:  0.01250363
## ***************************
## 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
##          1.951952          -0.005406           0.178816           0.093900          -0.011207
##      pos:log_freq
##          0.064137
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4404 Residual
## Null Deviance:       2370
## Residual Deviance: 2319  AIC: 2390
## log likelihood:  -1159.369
## Nagelkerke R2:  0.02785828
## % pres/err predicted correctly:  -616.7087
## % of predictable range [ (model-null)/(1-null) ]:  0.01232828
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.09933     -0.01803      0.15262      0.13478
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2324  AIC: 2390
## log likelihood:  -1162.068
## Nagelkerke R2:  0.02494683
## % pres/err predicted correctly:  -617.5608
## % of predictable range [ (model-null)/(1-null) ]:  0.01096587
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##        2.1011727        -0.0217760         0.3851219        -0.0001176         0.1536149
## stimlen:log_freq
##       -0.0325654
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4404 Residual
## Null Deviance:       2370
## Residual Deviance: 2321  AIC: 2391
## log likelihood:  -1160.447
## Nagelkerke R2:  0.02669555
## % pres/err predicted correctly:  -617.0679
## % of predictable range [ (model-null)/(1-null) ]:  0.01175403
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           2.02793           -0.01034           -0.00483            0.17690            0.09086
## I(pos^2):log_freq       pos:log_freq
##          -0.01115            0.06410
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4403 Residual
## Null Deviance:       2370
## Residual Deviance: 2319  AIC: 2392
## log likelihood:  -1159.331
## Nagelkerke R2:  0.02789933
## % pres/err predicted correctly:  -616.7345
## % of predictable range [ (model-null)/(1-null) ]:  0.01228696
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##   2.1102391   -0.0183388    0.0007518    0.1466346    0.1347725
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4405 Residual
## Null Deviance:       2370
## Residual Deviance: 2324  AIC: 2392
## log likelihood:  -1162.066
## Nagelkerke R2:  0.02494863
## % pres/err predicted correctly:  -617.568
## % of predictable range [ (model-null)/(1-null) ]:  0.01095431
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          2.041049          -0.015016           0.250838          -0.004253           0.176404
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.021678          -0.010095           0.061227
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4402 Residual
## Null Deviance:       2370
## Residual Deviance: 2317  AIC: 2392
## log likelihood:  -1158.694
## Nagelkerke R2:  0.02858567
## % pres/err predicted correctly:  -616.579
## % of predictable range [ (model-null)/(1-null) ]:  0.01253561
## ***************************
## 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)
##          0.24288           0.22171          -0.15618           1.52618           0.01957
##      stimlen:pos
##         -0.17545
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4404 Residual
## Null Deviance:       2370
## Residual Deviance: 2335  AIC: 2403
## log likelihood:  -1167.643
## Nagelkerke R2:  0.01892059
## % pres/err predicted correctly:  -619.5869
## % of predictable range [ (model-null)/(1-null) ]:  0.007726202
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.4076      -0.0611       0.1519
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2343  AIC: 2404
## log likelihood:  -1171.486
## Nagelkerke R2:  0.01475779
## % pres/err predicted correctly:  -620.6666
## % of predictable range [ (model-null)/(1-null) ]:  0.005999847
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.76303      0.01747      0.38968     -0.02810
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2341  AIC: 2404
## log likelihood:  -1170.388
## Nagelkerke R2:  0.01594777
## % pres/err predicted correctly:  -620.2758
## % of predictable range [ (model-null)/(1-null) ]:  0.006624768
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.9979       0.1354
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2405
## log likelihood:  -1173
## Nagelkerke R2:  0.01311572
## % pres/err predicted correctly:  -621.03
## % of predictable range [ (model-null)/(1-null) ]:  0.005418915
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.423565    -0.061542     0.001101     0.143092
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2343  AIC: 2406
## log likelihood:  -1171.483
## Nagelkerke R2:  0.01476171
## % pres/err predicted correctly:  -620.6775
## % of predictable range [ (model-null)/(1-null) ]:  0.005982464
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.971113    -0.002281     0.153813
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2407
## log likelihood:  -1172.985
## Nagelkerke R2:  0.01313287
## % pres/err predicted correctly:  -620.9994
## % of predictable range [ (model-null)/(1-null) ]:  0.005467733
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          2.14655           0.04391           0.38338          -0.03244
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2348  AIC: 2415
## log likelihood:  -1174.082
## Nagelkerke R2:  0.01194249
## % pres/err predicted correctly:  -621.1744
## % of predictable range [ (model-null)/(1-null) ]:  0.00518803
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.14254      0.04772      0.13387
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2351  AIC: 2416
## log likelihood:  -1175.696
## Nagelkerke R2:  0.010189
## % pres/err predicted correctly:  -621.7814
## % of predictable range [ (model-null)/(1-null) ]:  0.004217458
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.484
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4409 Residual
## Null Deviance:       2370
## Residual Deviance: 2370  AIC: 2428
## log likelihood:  -1185.057
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -624.419
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  15
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    2.448753     0.004595
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2370  AIC: 2429
## log likelihood:  -1185.048
## Nagelkerke R2:  1.029884e-05
## % pres/err predicted correctly:  -624.4153
## % of predictable range [ (model-null)/(1-null) ]:  5.940735e-06
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	[illegible]	I(pos^2)	[illegible]	log_freq:I(pos^2)	[illegible]	stimlen:I(pos^2)
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.2247232	NA	0.1350509	-0.0258610	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.1389448	NA	0.1483189	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0218241	0.3850807	-0.0325604	0.1528798	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0127592	0.2201851	NA	0.1384076	-0.0253771	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0176327	0.3963456	-0.0257396	0.1428798	NA	-0.018940	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	I(pos^2):log_freq	stimlen:pos	stimlen:I(pos^2)
preserved ~ (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	[illegible]	[illegible]	-0.0054[illegible]	-0.0051[illegible]	NA	NA	NA
preserved ~ stimlen + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0180332	[illegible]	NA	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0217760	0.3851219	-0.0325654	[illegible]	NA	-0.0001176	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0103366	[illegible]	NA	[illegible]	[illegible]	-0.0048[illegible]	-0.0011517	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0183388	[illegible]	NA	[illegible]	NA	0.0007518	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0150163	0.2508377	-0.0216776	[illegible]	0.061227	-0.0042528	NA	-0.0100954	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	NA	-0.1561752	NA	NA	-0.1754481	0.0195737
preserved ~ stimlen + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0610988	NA	NA	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	NA	NA	NA	NA	-0.0281045	NA
preserved ~ pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0615424	NA	NA	[illegible]	NA	0.0011[illegible]	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	NA	-0.0022809	NA	NA	NA	NA
preserved ~ stimlen * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0324407	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAICexpAICwNagR2(Intercept)stimlenlog_freqstimlenpos_freqlog_freqI(pos^2)pos^2)log_freqstimlenstimlen:I(pos^2)									
preserved ~ stimlen	2429.4179787710000000002143753445949	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)          pos      log_freq  pos:log_freq
##      2.01550      0.13505       0.22472      -0.02586
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2321  AIC: 2388
```

```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
```

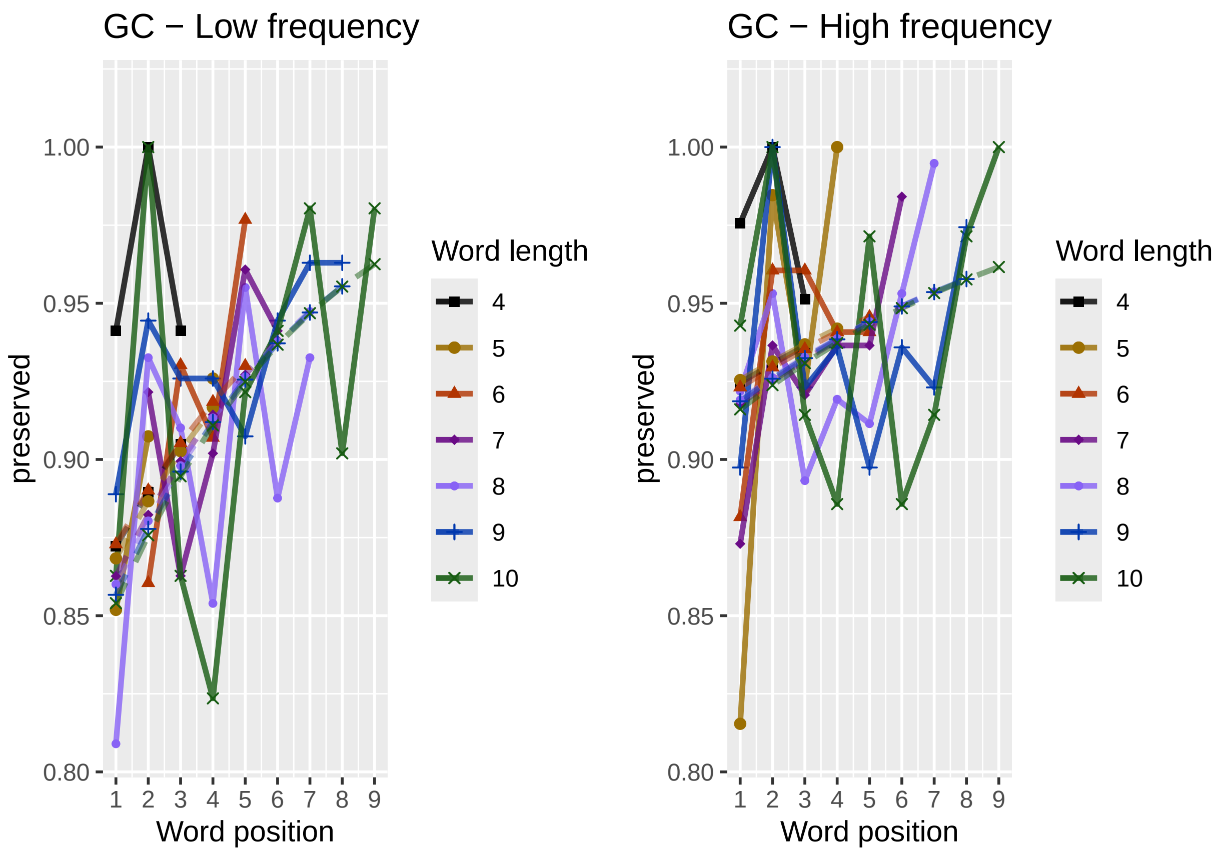
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale 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 3 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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      CumPres
##      2.0207       0.2067
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2323  AIC: 2380
## log likelihood:  -1161.305
## Nagelkerke R2:  0.02576994
## % pres/err predicted correctly:  -617.8757
## % of predictable range [ (model-null)/(1-null) ]:  0.01046233
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.6276      -0.4384
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2343  AIC: 2400
## log likelihood:  -1171.352
## Nagelkerke R2:  0.01490341
## % pres/err predicted correctly:  -618.8728
## % of predictable range [ (model-null)/(1-null) ]:  0.008868108
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.9979       0.1354
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2405
## log likelihood:  -1173
## Nagelkerke R2:  0.01311572
## % pres/err predicted correctly:  -621.03
## % of predictable range [ (model-null)/(1-null) ]:  0.005418915
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.971113    -0.002281     0.153813
```

```
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2407
## log likelihood:  -1172.985
## Nagelkerke R2:  0.01313287
## % pres/err predicted correctly:  -620.9994
## % of predictable range [ (model-null)/(1-null) ]:  0.005467733
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.484
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4409 Residual
## Null Deviance:       2370
## Residual Deviance: 2370  AIC: 2428
## log likelihood:  -1185.057
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -624.419
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    2.448753     0.004595
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2370  AIC: 2429
## log likelihood:  -1185.048
## Nagelkerke R2:  1.029884e-05
## % pres/err predicted correctly:  -624.4153
## % of predictable range [ (model-null)/(1-null) ]:  5.940735e-06
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	2379.760	0.00000	1.00e+00	0.9999442	0.0257692	2.020746	0.2066787	NA	NA	NA	NA
preserved ~ CumErr	2399.543	19.78353	5.06e-05	0.0000506	0.0149034	1.627556	NA	-0.4383797	NA	NA	NA
preserved ~ pos	2404.686	24.92571	3.90e-06	0.0000039	0.0131151	1.997935	NA	NA	NA	0.1354309	NA
preserved ~ (I(pos^2) + pos)	2406.735	26.97526	1.40e-06	0.0000014	0.0131329	1.971113	NA	NA	-0.0022809	0.1538133	NA
preserved ~ 1	2427.521	47.76077	0.00e+00	0.0000000	0.0000002	2.484047	NA	NA	NA	NA	NA
preserved ~ stimlen	2429.479	49.71958	0.00e+00	0.0000000	0.0000103	2.448753	NA	NA	NA	NA	0.0045949

```
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	0.8974800	582
O	0.8997377	2033
P	0.8055556	36
S	0.9108073	256
V	0.9692837	1503

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

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.9916       0.2536
##
## Degrees of Freedom: 4117 Total (i.e. Null);  4116 Residual
## Null Deviance:       2178
## Residual Deviance: 2120  AIC: 2174
## log likelihood:  -1059.787
## Nagelkerke R2:  0.03406291
## % pres/err predicted correctly:  -562.7773
## % of predictable range [ (model-null)/(1-null) ]:  0.013785
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.9555       0.1549
##
## Degrees of Freedom: 4117 Total (i.e. Null);  4116 Residual
## Null Deviance:       2178
## Residual Deviance: 2149  AIC: 2205
## log likelihood:  -1074.513
## Nagelkerke R2:  0.01683099
## % pres/err predicted correctly:  -566.6978
## % of predictable range [ (model-null)/(1-null) ]:  0.006926801
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##   1.9524175   -0.0002702    0.1570339
##
## Degrees of Freedom: 4117 Total (i.e. Null);  4115 Residual
## Null Deviance:       2178
## Residual Deviance: 2149  AIC: 2207
## log likelihood:  -1074.512
## Nagelkerke R2:  0.01683122
## % pres/err predicted correctly:  -566.6946
## % of predictable range [ (model-null)/(1-null) ]:  0.00693254
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.6382      -0.4478
##
## Degrees of Freedom: 4117 Total (i.e. Null);  4116 Residual
## Null Deviance:       2178
## Residual Deviance: 2156  AIC: 2209
## log likelihood:  -1077.794
## Nagelkerke R2:  0.01297459
## % pres/err predicted correctly:  -566.4445
## % of predictable range [ (model-null)/(1-null) ]:  0.007370051
## ***************************
## model index:  6
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.509
##
## Degrees of Freedom: 4117 Total (i.e. Null);  4117 Residual
## Null Deviance:       2178
## Residual Deviance: 2178  AIC: 2232
## log likelihood:  -1088.794
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -570.6576
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    2.464026     0.005912
##
## Degrees of Freedom: 4117 Total (i.e. Null);  4116 Residual
## Null Deviance:       2178
## Residual Deviance: 2178  AIC: 2234
## log likelihood:  -1088.78
## Nagelkerke R2:  1.69541e-05
## % pres/err predicted correctly:  -570.652
## % of predictable range [ (model-null)/(1-null) ]:  9.745277e-06
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	2173.554	0.00000	1e+00	0.9999997	0.0340621	1.991559	0.2536432	NA	NA	NA	NA
preserved ~ pos	2204.755	31.20131	2e-07	0.0000002	0.0168310	1.955516	NA	NA	NA	0.1548814	NA
preserved ~ (I(pos^2) + pos)	2206.762	33.20848	1e-07	0.0000001	0.0168311	1.952417	NA	NA	-0.0002702	0.1570339	NA
preserved ~ CumErr	2208.808	35.25422	0e+00	0.0000000	0.0129742	2.638234	NA	-0.4477844	NA	NA	NA
preserved ~ 1	2231.830	58.27634	0e+00	0.0000000	0.0000000	2.509450	NA	NA	NA	NA	NA
preserved ~ stimlen	2233.772	60.21813	0e+00	0.0000000	0.0000170	2.464026	NA	NA	NA	NA	0.0059116

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

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0091       0.3447
##
## Degrees of Freedom: 3535 Total (i.e. Null);  3534 Residual
## Null Deviance:       1793
## Residual Deviance: 1730  AIC: 1767
## log likelihood:  -865.1274
## Nagelkerke R2:  0.04415313
## % pres/err predicted correctly:  -454.8367
## % of predictable range [ (model-null)/(1-null) ]:  0.01747111
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.9618       0.1758
##
## Degrees of Freedom: 3535 Total (i.e. Null);  3534 Residual
## Null Deviance:       1793
## Residual Deviance: 1762  AIC: 1800
## log likelihood:  -880.9199
## Nagelkerke R2:  0.02198993
## % pres/err predicted correctly:  -458.8284
## % of predictable range [ (model-null)/(1-null) ]:  0.008867309
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.855926    -0.009847     0.253203
##
## Degrees of Freedom: 3535 Total (i.e. Null);  3533 Residual
## Null Deviance:       1793
## Residual Deviance: 1761  AIC: 1802
## log likelihood:  -880.6971
## Nagelkerke R2:  0.02230399
## % pres/err predicted correctly:  -458.6454
## % of predictable range [ (model-null)/(1-null) ]:  0.009261632
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.6836      -0.4594
##
## Degrees of Freedom: 3535 Total (i.e. Null);  3534 Residual
## Null Deviance:       1793
## Residual Deviance: 1778  AIC: 1815
## log likelihood:  -889.1652
## Nagelkerke R2:  0.01033959
## % pres/err predicted correctly:  -460.4791
## % of predictable range [ (model-null)/(1-null) ]:  0.005309388
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.576
##
## Degrees of Freedom: 3535 Total (i.e. Null);  3535 Residual
## Null Deviance:       1793
## Residual Deviance: 1793  AIC: 1829
## log likelihood:  -896.4508
## Nagelkerke R2:  -5.582858e-16
## % pres/err predicted correctly:  -462.9423
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.67194     -0.01248
##
## Degrees of Freedom: 3535 Total (i.e. Null);  3534 Residual
## Null Deviance:       1793
## Residual Deviance: 1793  AIC: 1831
## log likelihood:  -896.3986
## Nagelkerke R2:  7.422689e-05
## % pres/err predicted correctly:  -462.9306
## % of predictable range [ (model-null)/(1-null) ]:  2.517587e-05
## ***************************
```

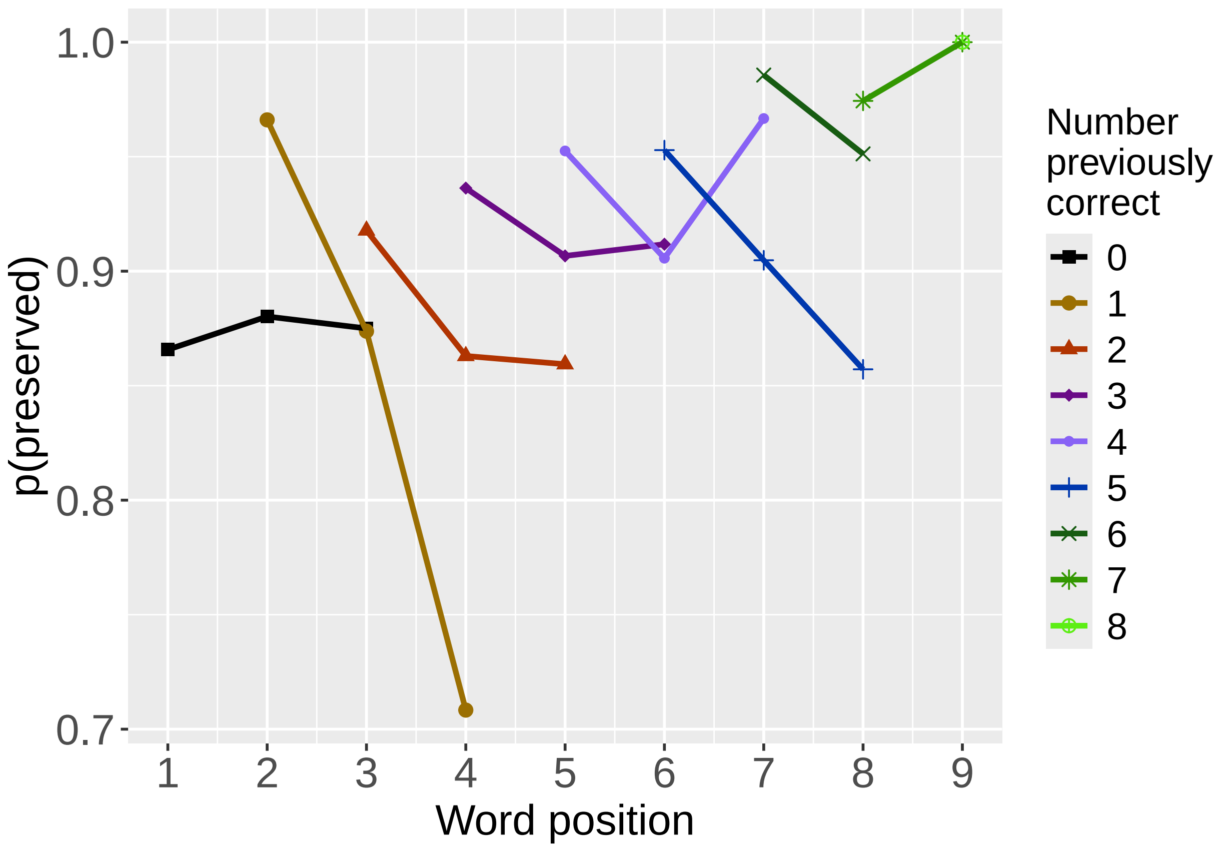
```r
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	1766.702	0.00000	1e+00	0.9999999	0.0441532	1.009078	0.3447131	NA	NA	NA	NA
preserved ~ pos	1799.858	33.15523	1e-07	0.0000001	0.0219899	1.961796	NA	NA	NA	0.1757621	NA
preserved ~ (I(pos^2) + pos)	1801.580	34.87738	0e+00	0.0000000	0.0223041	1.855926	NA	NA	-0.0098473	0.2532032	NA
preserved ~ CumErr	1815.266	48.56351	0e+00	0.0000000	0.0103393	2.683581	NA	-0.4594472	NA	NA	NA
preserved ~ 1	1829.146	62.44334	0e+00	0.0000000	0.0000000	2.575965	NA	NA	NA	NA	NA
preserved ~ stimlen	1831.073	64.37021	0e+00	0.0000000	0.0000742	2.671937	NA	NA	NA	NA	-0.0124833

```r
# 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.
```

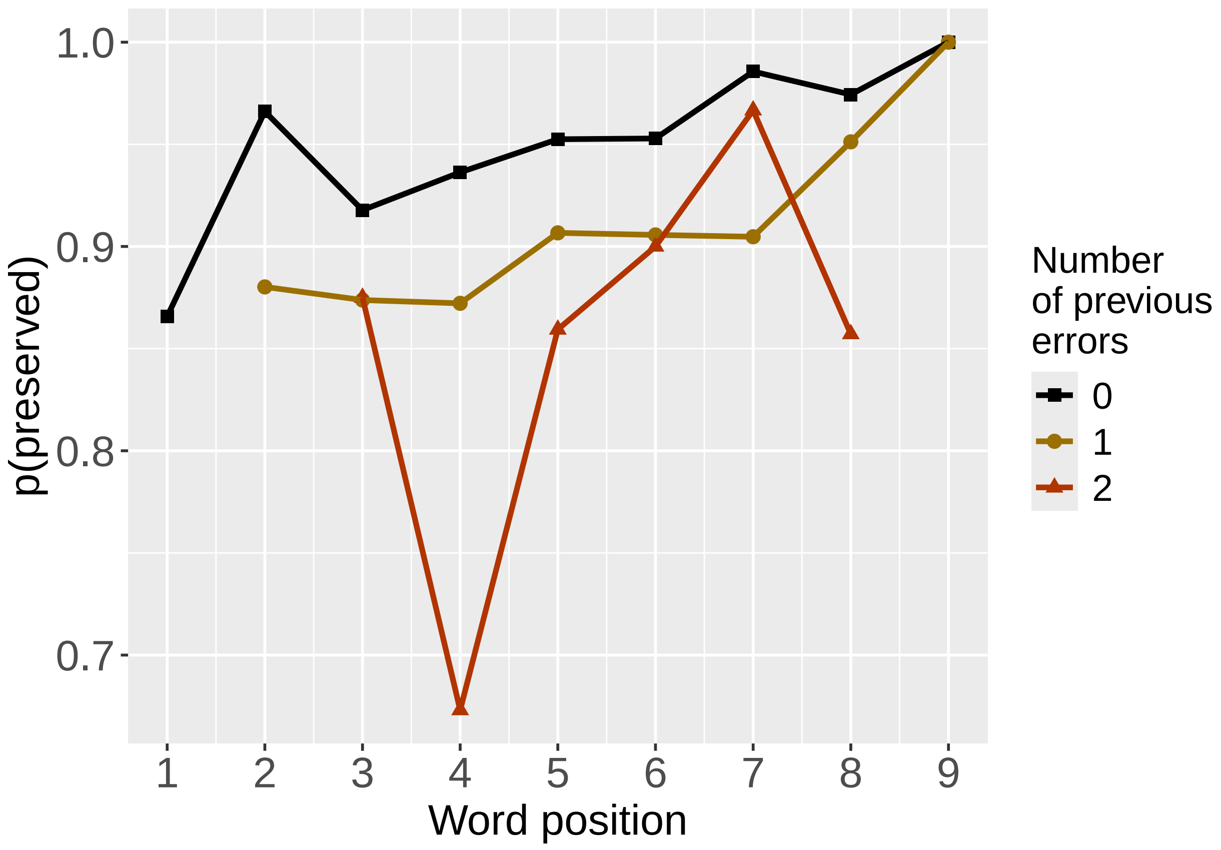
```r
print(PrevCorPlot)
```



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

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

```r
print(PrevErrPlot)
```



```r
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
```

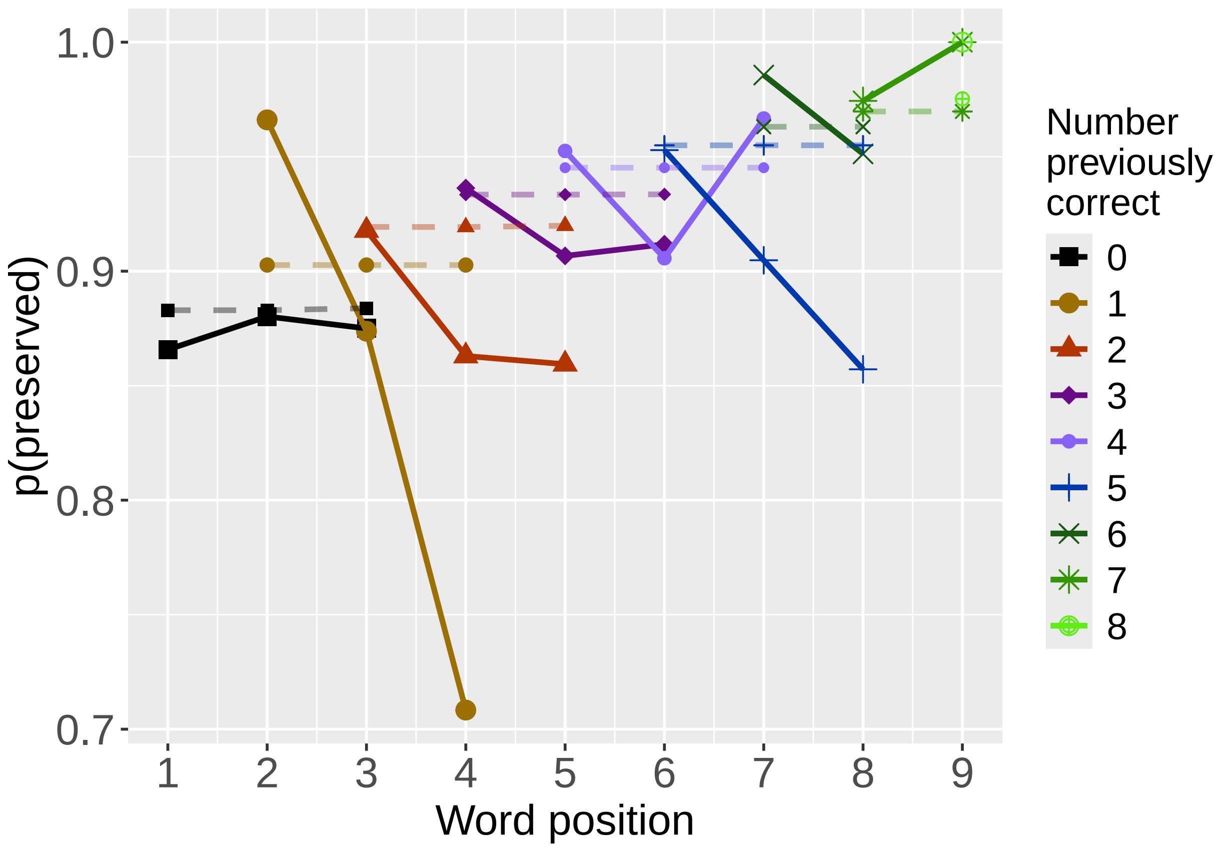
```r
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

```r
# 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.
```

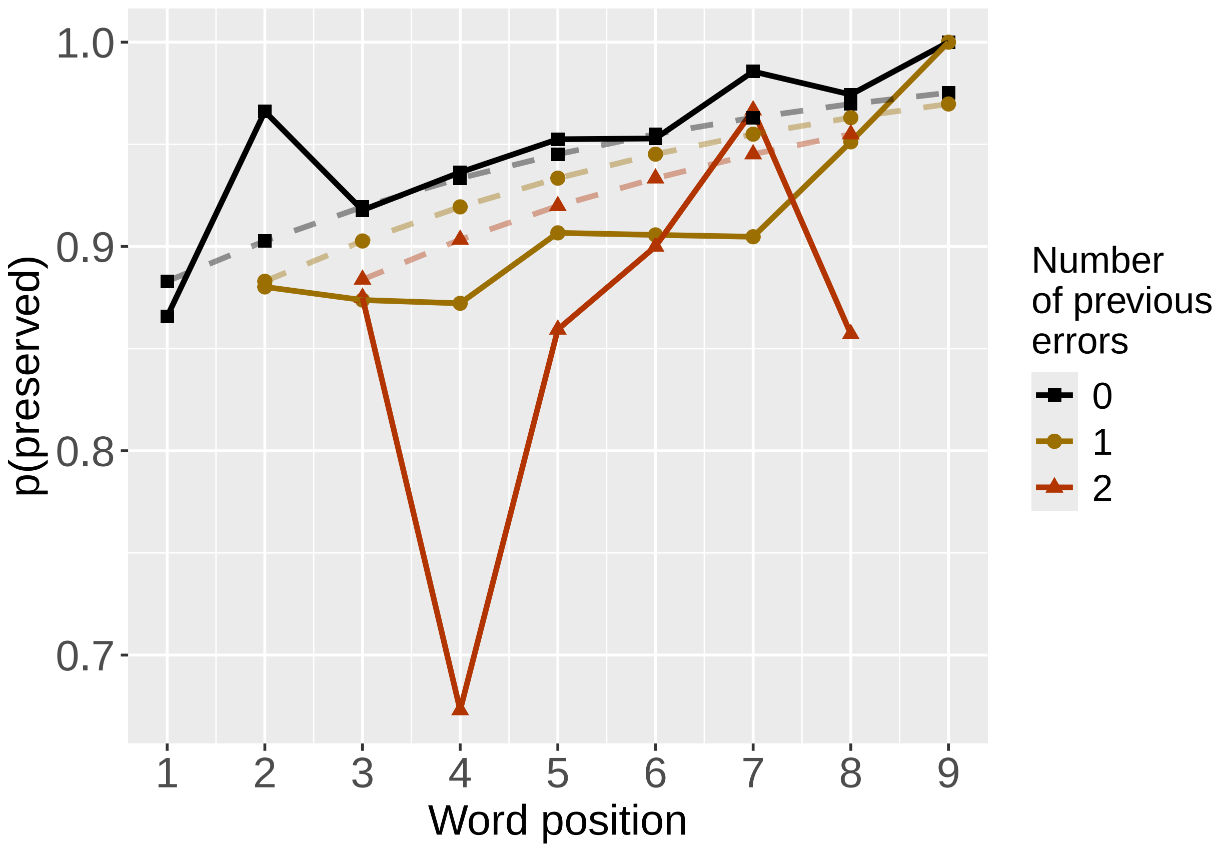
```r
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos
##    2.594482     0.706705    -0.002723    -0.461300
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2290  AIC: 2348
## log likelihood:  -1144.965
## Nagelkerke R2:  0.04333777
## % pres/err predicted correctly:  -611.4042
## % of predictable range [ (model-null)/(1-null) ]:  0.02080986
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0207       0.2067
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2323  AIC: 2380
## log likelihood:  -1161.305
## Nagelkerke R2:  0.02576994
## % pres/err predicted correctly:  -617.8757
## % of predictable range [ (model-null)/(1-null) ]:  0.01046233
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.971113    -0.002281     0.153813
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2407
## log likelihood:  -1172.985
## Nagelkerke R2:  0.01313287
## % pres/err predicted correctly:  -620.9994
## % of predictable range [ (model-null)/(1-null) ]:  0.005467733
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos
preserved ~ CumPres + I(pos^2) + pos	2348.189	0.00000	1e+00	0.9999999	0.0433378	2.594482	0.7067045	-0.0027231	-0.4613001

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos
preserved ~ CumPres	2379.760	31.57102	1e-07	0.0000001	0.0257699	2.020746	0.2066787	NA	NA
preserved ~ I(pos^2) + pos	2406.735	58.54628	0e+00	0.0000000	0.0131329	1.971113	NA	-0.0022809	0.1538133

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres      stimlen
##     2.59873      0.22848     -0.08182
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2317  AIC: 2376
## log likelihood:  -1158.544
## Nagelkerke R2:  0.02874787
## % pres/err predicted correctly:  -617.2071
## % of predictable range [ (model-null)/(1-null) ]:  0.01153142
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0207       0.2067
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2323  AIC: 2380
## log likelihood:  -1161.305
## Nagelkerke R2:  0.02576994
## % pres/err predicted correctly:  -617.8757
## % of predictable range [ (model-null)/(1-null) ]:  0.01046233
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    2.448753     0.004595
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2370  AIC: 2429
## log likelihood:  -1185.048
## Nagelkerke R2:  1.029884e-05
## % pres/err predicted correctly:  -624.4153
## % of predictable range [ (model-null)/(1-null) ]:  5.940735e-06
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	stimlen
preserved ~ CumPres + stimlen	2376.375	0.000000	1.0000000	0.8445173	0.0287479	2.598730	0.2284768	-0.0818197
preserved ~ CumPres	2379.760	3.384462	0.1841083	0.1554827	0.0257699	2.020746	0.2066787	NA
preserved ~ stimlen	2429.479	53.104046	0.0000000	0.0000000	0.0000103	2.448753	NA	0.0045949

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"pos"

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos
##      2.6265       0.7071      -0.4835
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2290  AIC: 2346
## log likelihood:  -1144.987
## Nagelkerke R2:  0.04331442
## % pres/err predicted correctly:  -611.4624
## % of predictable range [ (model-null)/(1-null) ]:  0.02071668
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0207       0.2067
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2323  AIC: 2380
## log likelihood:  -1161.305
## Nagelkerke R2:  0.02576994
## % pres/err predicted correctly:  -617.8757
## % of predictable range [ (model-null)/(1-null) ]:  0.01046233
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.9979       0.1354
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2346  AIC: 2405
## log likelihood:  -1173
## Nagelkerke R2:  0.01311572
## % pres/err predicted correctly:  -621.03
## % of predictable range [ (model-null)/(1-null) ]:  0.005418915
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	pos
preserved ~ CumPres + pos	2346.134	0.00000	1	1	0.0433144	2.626542	0.7071145	-0.4834870
preserved ~ CumPres	2379.760	33.62627	0	0	0.0257699	2.020746	0.2066787	NA
preserved ~ pos	2404.686	58.55199	0	0	0.0131157	1.997935	NA	0.1354309

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos	stimlen
preserved ~ CumPres + pos	2346.134	0.000000	1.000000	1.000000	0.0433144	2.626542	0.7071145	NA	-0.4834870	NA
preserved ~ CumPres + I(pos^2) + pos	2348.189	0.000000	1.000000	0.999999	0.0433378	2.594482	0.7067045	-0.0027231	-0.4613001	NA
preserved ~ CumPres + stimlen	2376.375	0.000000	1.000000	0.844517	0.0287479	2.598730	0.2284768	NA	NA	-0.0818197
preserved ~ CumPres	2379.760	31.571019	0.000000	1.000000	0.0257699	2.020746	0.2066787	NA	NA	NA
preserved ~ CumPres	2379.760	3.384462	0.184108	0.155482	0.0257699	2.020746	0.2066787	NA	NA	NA
preserved ~ CumPres	2379.760	33.626275	0.000000	0.000000	0.0257699	2.020746	0.2066787	NA	NA	NA
preserved ~ pos	2404.686	58.551986	0.000000	0.000000	0.0131157	1.997935	NA	NA	0.1354309	NA
preserved ~ I(pos^2) + pos	2406.735	58.546277	0.000000	0.000000	0.0131329	1.971113	NA	-0.0022809	0.1538133	NA
preserved ~ stimlen	2429.479	53.104046	0.000000	0.000000	0.0000103	2.448753	NA	NA	NA	0.0045949

```
# explore influence of frequency and length

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

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

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

## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos     log_freq
##      2.5682       0.6633      -0.4341       0.1128
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2276  AIC: 2337
## log likelihood:  -1137.856
## Nagelkerke R2:  0.05094042
## % pres/err predicted correctly:  -609.5931
## % of predictable range [ (model-null)/(1-null) ]:  0.02370558
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos      stimlen     log_freq
##     2.80025      0.66804     -0.43029     -0.03361      0.10512
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4405 Residual
## Null Deviance:       2370
## Residual Deviance: 2275  AIC: 2338
## log likelihood:  -1137.447
## Nagelkerke R2:  0.05137688
## % pres/err predicted correctly:  -609.4866
## % of predictable range [ (model-null)/(1-null) ]:  0.02387592
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos      stimlen
##     3.08348      0.71049     -0.46847     -0.06761
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2286  AIC: 2345
## log likelihood:  -1143.178
## Nagelkerke R2:  0.04525165
## % pres/err predicted correctly:  -610.9323
## % of predictable range [ (model-null)/(1-null) ]:  0.02156435
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos
##      2.6265       0.7071      -0.4835
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2290  AIC: 2346
## log likelihood:  -1144.987
## Nagelkerke R2:  0.04331442
## % pres/err predicted correctly:  -611.4624
## % of predictable range [ (model-null)/(1-null) ]:  0.02071668
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.484
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4409 Residual
## Null Deviance:       2370
## Residual Deviance: 2370  AIC: 2428
## log likelihood:  -1185.057
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -624.419
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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))
```

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	pos	log_freq	stimlen
preserved ~ CumPres + pos + log_freq	2337.105	0.000000	1.0000000	0.6167205	0.0509404	1.568233	0.6632719	-0.4340773	0.1128163	NA
preserved ~ CumPres + pos + stimlen + log_freq	2338.166	1.061166	0.5882619	0.3627932	0.0513762	2.800247	0.6680417	-0.4302916	0.1051201	-0.0336137
preserved ~ CumPres + pos + stimlen	2344.715	7.609398	0.0222659	0.0137318	0.0452513	3.083477	0.7104934	-0.4684694	NA	-0.0676100
preserved ~ CumPres + pos	2346.134	9.028425	0.0109522	0.0067545	0.0433144	1.626542	0.7071145	-0.4834870	NA	NA
preserved ~ 1	2427.521	90.415470	0.0000000	0.0000000	0.0000000	2.484047	NA	NA	NA	NA

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

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

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

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

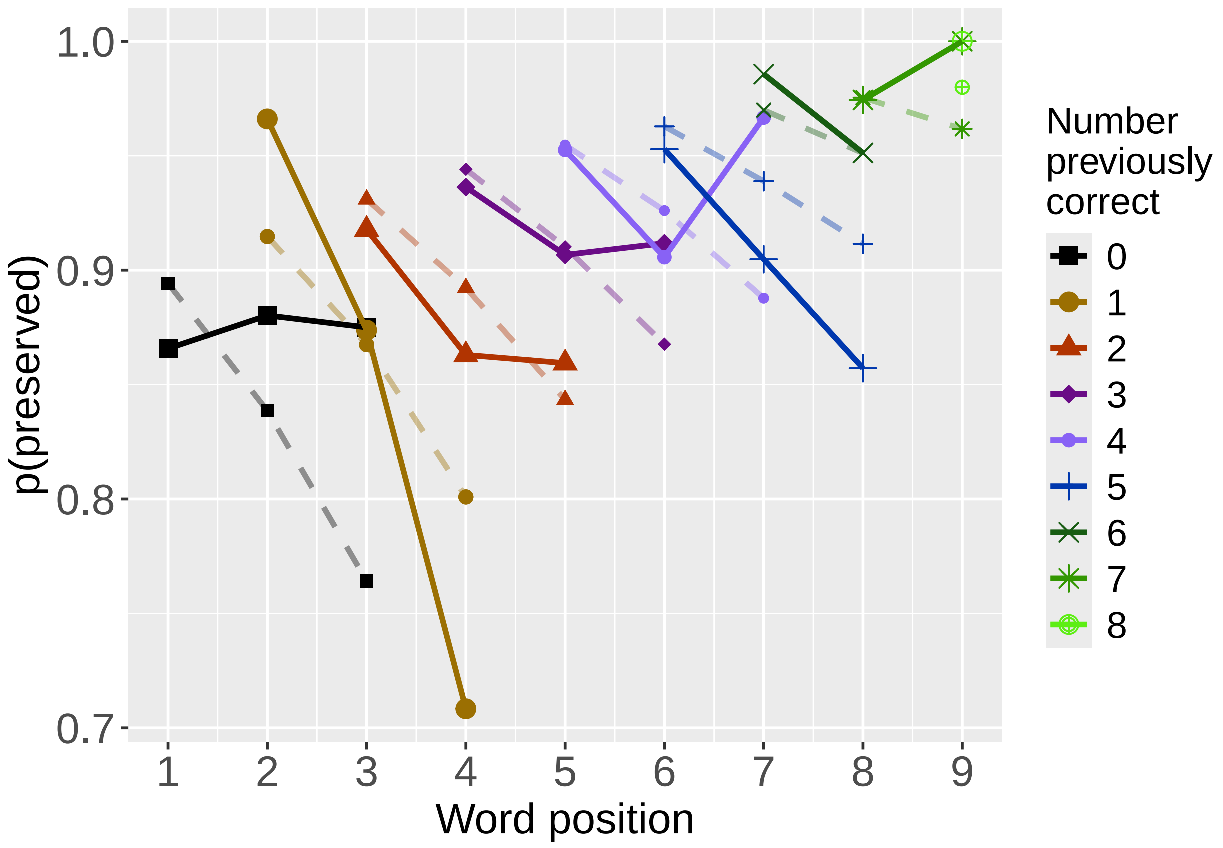
```
## Single term deletions
##
## Model:
## preserved ~ CumPres + pos + log_freq
##          Df Deviance    AIC
## CumPres   1   2324.4 2383.8
## pos       1   2301.6 2360.9
## log_freq  1   2290.0 2349.4
## <none>        2275.7 2337.1
```

```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.
```

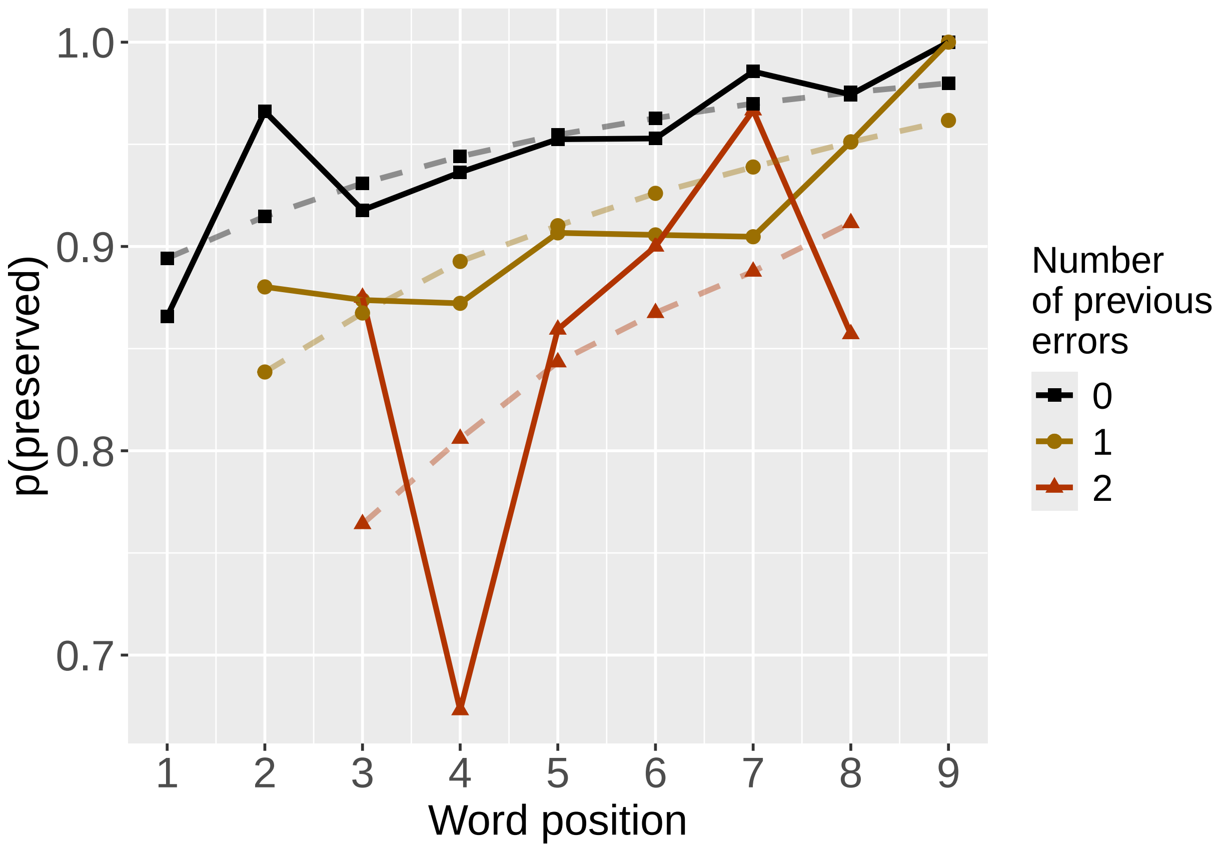
```
print(PrevCorPlot)
```



```
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),
```

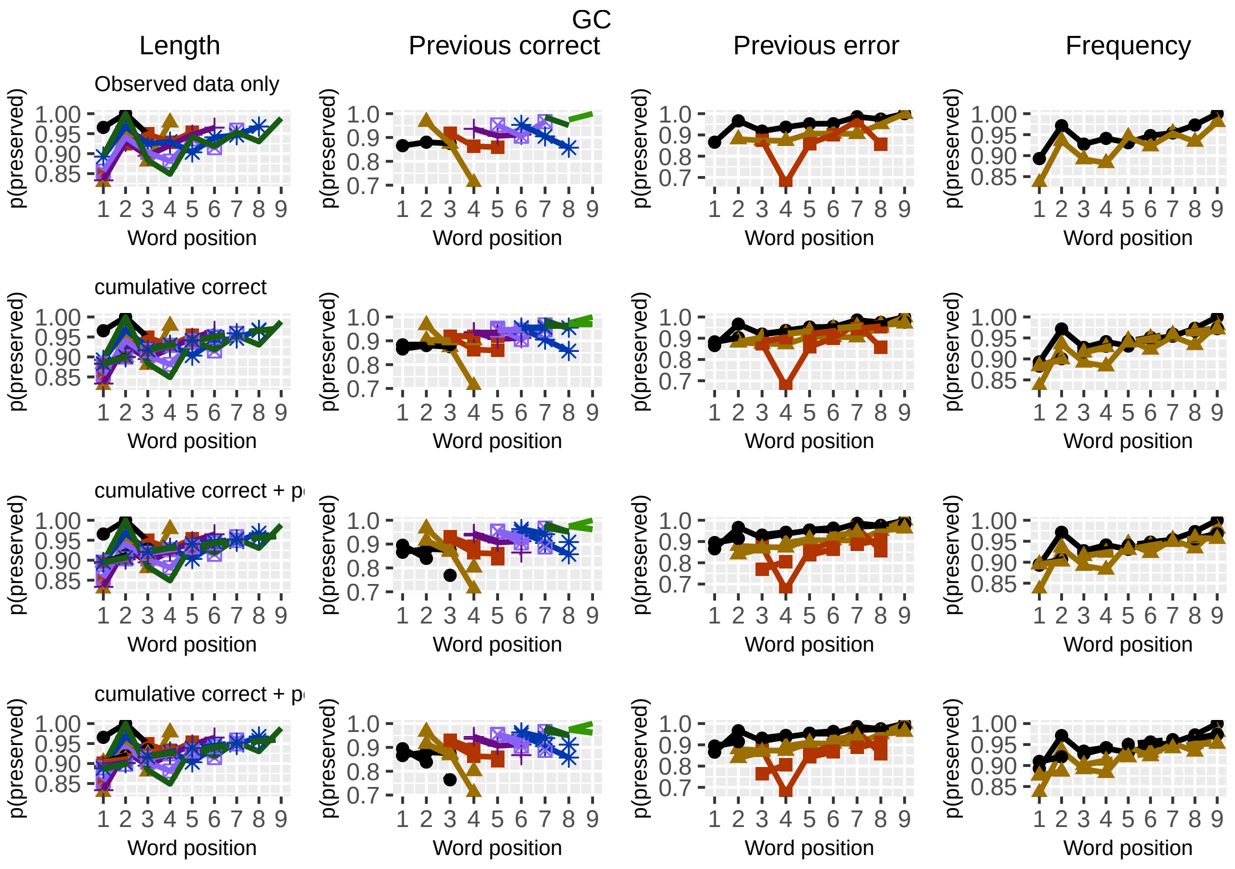
```r
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random 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)
}
```

```r
# 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!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0207       0.2067
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4408 Residual
## Null Deviance:       2370
## Residual Deviance: 2323  AIC: 2380
## log likelihood:  -1161.305
## Nagelkerke R2:  0.02576994
## % pres/err predicted correctly:  -617.8757
## % of predictable range [ (model-null)/(1-null) ]:  0.01046233
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos
##      2.6265       0.7071      -0.4835
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4407 Residual
## Null Deviance:       2370
## Residual Deviance: 2290  AIC: 2346
## log likelihood:  -1144.987
## Nagelkerke R2:  0.04331442
## % pres/err predicted correctly:  -611.4624
## % of predictable range [ (model-null)/(1-null) ]:  0.02071668
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos     log_freq
##      2.5682       0.6633      -0.4341       0.1128
##
## Degrees of Freedom: 4409 Total (i.e. Null);  4406 Residual
## Null Deviance:       2370
## Residual Deviance: 2276  AIC: 2337
## log likelihood:  -1137.856
## Nagelkerke R2:  0.05094042
## % pres/err predicted correctly:  -609.5931
## % of predictable range [ (model-null)/(1-null) ]:  0.02370558
## ***************************
## `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()`).
```

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



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

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

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

```r
kable(DAContributionAverage)
```

	CumPres	pos	log_freq
McFadden	0.0220523	0.0118725	0.0058256
SquaredCorrelation	0.0119974	0.0064567	0.0031716
Nagelkerke	0.0119974	0.0064567	0.0031716
Estrella	0.0122392	0.0065917	0.0032315

	deviance	deviance_explained
CumPres + pos + log_freq	2275.712	94.40223
CumPres + pos	2289.974	80.14055
CumPres	2322.610	47.50400
null	2370.114	0.00000

```
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!
```

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

```
##                                             model deviance deviance_explained
## CumPres + pos + log_freq CumPres + pos + log_freq 2275.712           94.40223
## CumPres + pos                       CumPres + pos 2289.974           80.14055
## CumPres                                   CumPres 2322.610           47.50400
## null                                         null 2370.114            0.00000
##                          percent_explained percent_of_explained_deviance
## CumPres + pos + log_freq          3.983025                     100.00000
## CumPres + pos                     3.381295                      84.89265
## CumPres                           2.004292                      50.32084
## null                              0.000000                            NA
##                          increment_in_explained
## CumPres + pos + log_freq               15.10735
## CumPres + pos                          34.57180
## CumPres                                50.32084
## null                                    0.00000
```

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

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

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumPres + pos + log_freq	3.983025	100.00000	15.10735
CumPres + pos	3.381295	84.89265	34.57180
CumPres	2.004291	50.32084	50.32084
null	0.000000	NA	0.00000

```
NagContributions <- t(DAContributionAverage[3,])
NagTotal<-sum(NagContributions)
NagPercents <- NagContributions / NagTotal
NagResultTable <- data.frame(NagContributions = NagContributions, NagPercents = NagPercents)
names(NagResultTable)<-c("NagContributions","NagPercents")
write.csv(NagResultTable,paste0(TablesDir,CurPat,"_",CurTask,"_nagelkerke_contributions_table.csv"),row.names = TRUE)
NagPercents
```

```
##          Nagelkerke
## CumPres   0.5547763
## pos       0.2985651
## log_freq  0.1466585
```

```
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
```

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

```
##                              model p_accounted_for model_deviance diff_CumPres
## 1               preserved ~ CumPres       0.5079112       2322.610    0.0000000
## 2           preserved ~ CumPres+pos       0.6441795       2289.974    0.1362684
## 3 preserved ~ CumPres+pos+log_freq       0.6500610       2275.712    0.1421499
```

model	p_accounted_for	model_deviance
preserved ~ CumPres	0.5079112	2322.610
preserved ~ CumPres+pos	0.6441795	2289.974
preserved ~ CumPres+pos+log_freq	0.6500610	2275.712

model	diff_CumPres	diff_CumPres+pos	diff_CumPres+pos+log_freq
preserved ~ CumPres	0.0000000	-0.1362684	-0.1421499
preserved ~ CumPres+pos	0.1362684	0.0000000	-0.0058815
preserved ~ CumPres+pos+log_freq	0.1421499	0.0058815	0.0000000

```
##   diff_CumPres+pos diff_CumPres+pos+log_freq
## 1     -0.136268358              -0.142149859
## 2      0.000000000              -0.005881502
## 3      0.005881502               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")
```