extras

Gabe and Jake

Tuesday, July 28, 2015

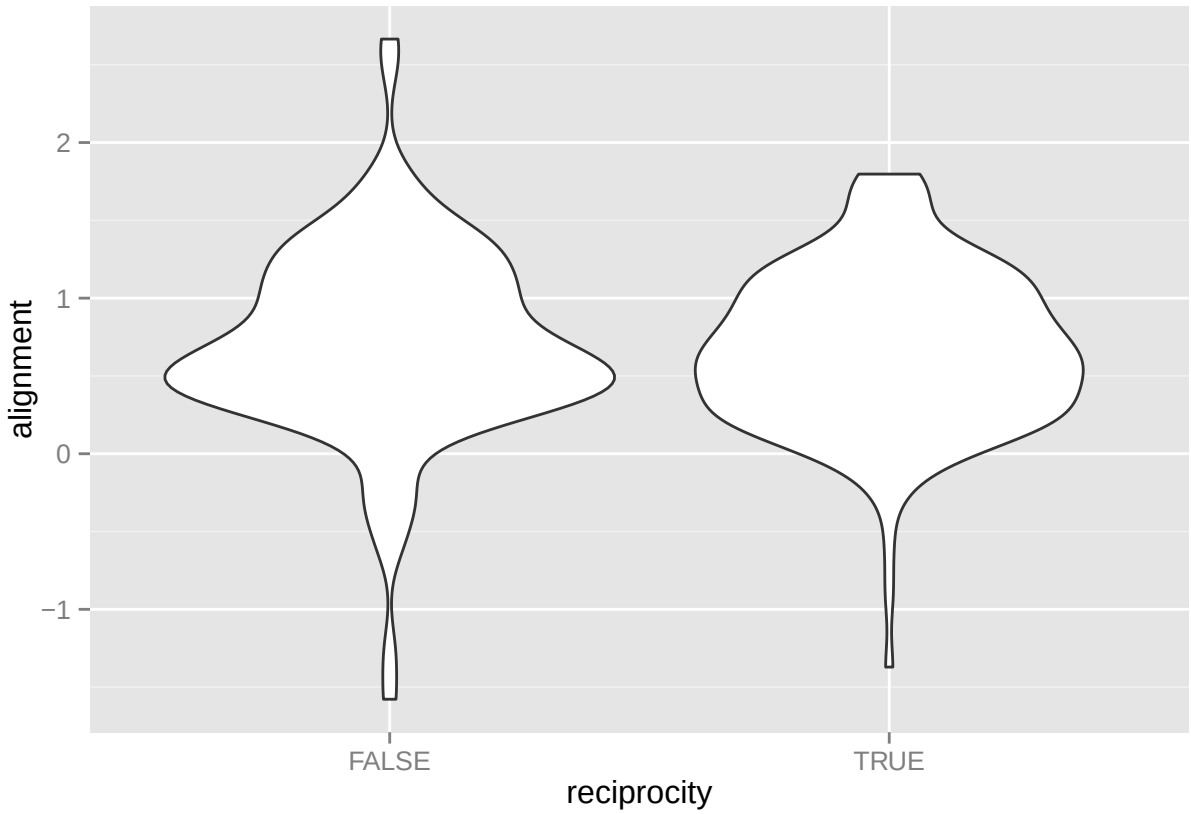
```
## Warning: package 'bit64' was built under R version 3.1.3

## Loading required package: bit
## Attaching package bit
## package:bit (c) 2008-2012 Jens Oehlschlaegel (GPL-2)
## creators: bit bitwhich
## coercion: as.logical as.integer as.bit as.bitwhich which
## operator: ! & | xor != ==
## querying: print length any all min max range sum summary
## bit access: length<- [ [<- [[ [[<-
## for more help type ?bit
##
## Attaching package: 'bit'
##
## The following object is masked from 'package:data.table':
##
##     setattr
##
## The following object is masked from 'package:base':
##
##     xor
##
## Attaching package bit64
## package:bit64 (c) 2011-2012 Jens Oehlschlaegel (GPL-2 with commercial restrictions)
## creators: integer64 seq :
## coercion: as.integer64 as.vector as.logical as.integer as.double as.character as.bin
## logical operator: ! & | xor != == < <= >= >
## arithmetic operator: + - * / %/% %% ^
## math: sign abs sqrt log log2 log10
## math: floor ceiling trunc round
## querying: is.integer64 is.vector [is.atomic} [length] is.na format print
## aggregation: any all min max range sum prod
## cumulation: diff cummin cummax cumsum cumprod
## access: length<- [ [<- [[ [[<-
## combine: c rep cbind rbind as.data.frame
## for more help type ?bit64
##
## Attaching package: 'bit64'
##
## The following object is masked from 'package:bit':
##
##     still.identical
##
## The following objects are masked from 'package:base':
##
##     :, %in%, is.double, match, order, rank

## Warning: package 'dplyr' was built under R version 3.1.3
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:data.table':
##
##     between, last
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
smoothalign <- function(df,sm,align="logodds") {
  if (align=="logodds") {
    return(log(df$ba+sm)-log(df$ba+df$nba+2*sm)-log(df$bna+sm)+log(df$bna+df$nbna+2*sm))
  } else if (align=="subodds") {
    return((df$ba+sm)/(df$ba+df$nba+2*sm)-(df$bna+sm)/(df$bna+df$nbna+2*sm))
  } else if (align=="logdnm") {
    return(log(df$ba+sm)-log(df$ba+df$nba+2*sm)-log(df$bna+df$ba+sm)+log(df$nba+df$ba+df$bna+df$nbna+2*s
  } else if (align=="subdnm") {
    return((df$ba+sm)/(df$ba+df$nba+2*sm)-(df$bna+df$ba+sm)/(df$nba+df$ba+df$bna+df$nbna+2*sm))
  } else {
    stop("Invalid alignment type.")
  }
}

d$lo1 <- smoothalign(d,1,"logodds")
d$sd0 <- smoothalign(d,0,"subdnm")

stopifnot(max(abs(d$lo1-d$pyalign))<.00001)
#stopifnot(max(abs(d$sd0-d$dnm))<.00001)
```

Reciprocity

```
library(dplyr)
library(ggplot2)

theme_set(theme_grey())

d2 <- d %>%
  filter((ba+nba)>5&(bna+nbna)>5) %>%
  group_by(vspeak,category, vreply, reciprocity) %>%
  summarize(convs=n(),alignment=mean(lo1), reciprocity=reciprocity)

ggplot(d2,aes(x=reciprocity,y=alignment)) + geom_violin()
```



Hmm, from the looks of the plot, it like reciprocity does have impact on alignment

Let's do a t-test

```
t.test(d2$alignment~d2$reciprocity)
```

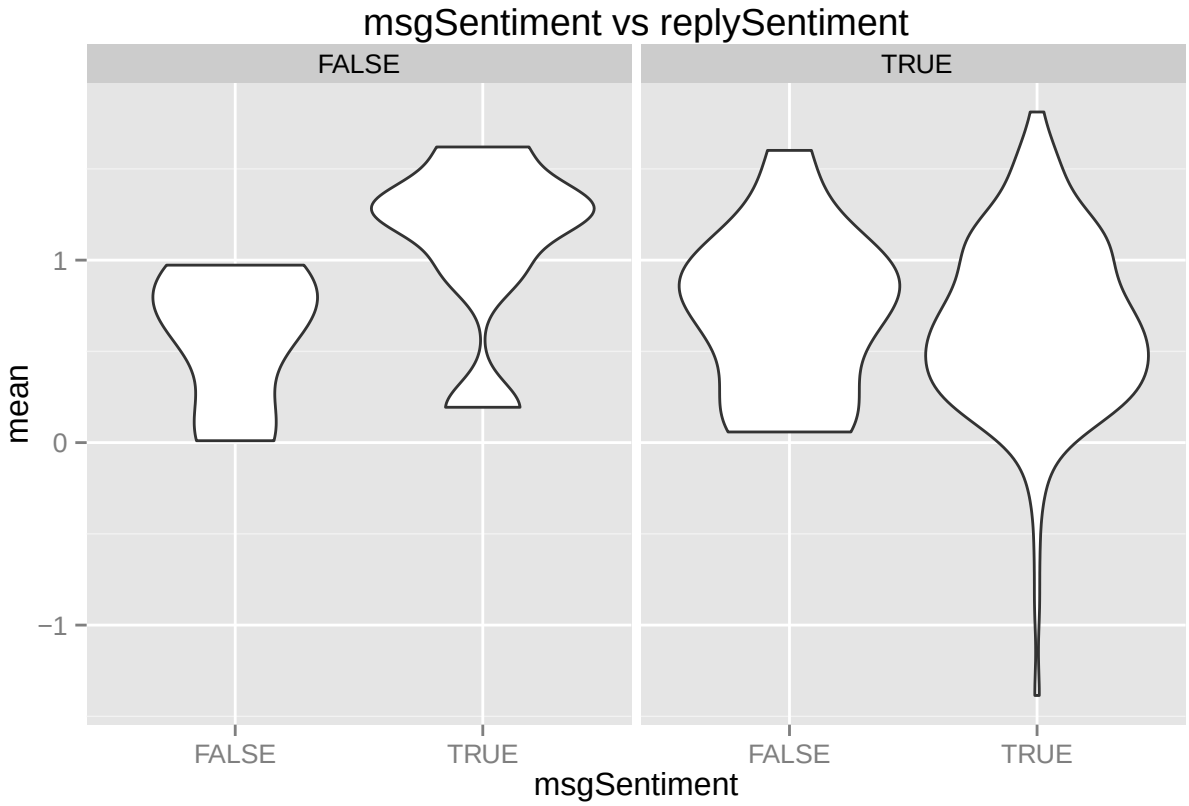
```
##
##  Welch Two Sample t-test
##
## data:  d2$alignment by d2$reciprocity
## t = 0.5148, df = 216.799, p-value = 0.6072
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.09928896  0.16949859
## sample estimates:
## mean in group FALSE  mean in group TRUE
##           0.7052213           0.6701165
```

Interesting..., The t-test gives us a p-value of 0.6, so there isn't much of an impact.

Sentiment

```
library(dplyr)
library(ggplot2)

theme_set(theme_grey())

d2 <- d %>%
  filter((ba+nba)>5&(bna+nbna)>5) %>%
```

```
  group_by(vspeak, vreply, category) %>%
  summarize(convs=n(),mean=mean(lo1), msgSentiment=(mean(msgSentiment)>0), replySentiment=(mean(replySe

ggplot(d2,aes(x=msgSentiment,y=mean)) + geom_violin() + facet_wrap(~replySentiment) + labs(title="msgSe
```



Clearly, there's an impact of sentiment on alignment. Let's do a t-test

```
d2 <- d %>%
  filter((ba+nba)>5&(bna+nbna)>5) %>%
  group_by(vspeak, vreply, category) %>%
  summarize(convs=n(),mean=mean(lo1), msgSentiment=(mean(msgSentiment)>0), replySentiment=(mean(replySe

t.test(d2$mean~d2$msgSentiment)
```

```
##
##  Welch Two Sample t-test
##
## data:  d2$mean by d2$msgSentiment
## t = 0.1054, df = 35.018, p-value = 0.9167
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1817409  0.2016381
## sample estimates:
## mean in group FALSE  mean in group TRUE
##           0.6710115           0.6610630
```

```
t.test(d2$mean~d2$replySentiment)
```

```
##
##  Welch Two Sample t-test
##
## data:  d2$mean by d2$replySentiment
## t = 1.2227, df = 13.778, p-value = 0.2419
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1329073  0.4841745
## sample estimates:
## mean in group FALSE  mean in group TRUE
##           0.8253305           0.6496969
```

Let's also do some manual checking of means and sparsity:

```
mean(filter(d2, msgSentiment==TRUE&replySentiment==TRUE)$mean)
```

```
## [1] 0.6431936
```

```
nrow(filter(d2, msgSentiment==TRUE&replySentiment==TRUE))
```

```
## [1] 148
```

```
mean(filter(d2, msgSentiment==TRUE&replySentiment==FALSE)$mean)
```

```
## [1] 1.10184
```

```
nrow(filter(d2, msgSentiment==TRUE&replySentiment==FALSE))
```

```
## [1] 6
```

```
mean(filter(d2, msgSentiment==FALSE&replySentiment==TRUE)$mean)
```

```
## [1] 0.7031683
```

```
nrow(filter(d2, msgSentiment==FALSE&replySentiment==TRUE))
```

```
## [1] 18
```

```
mean(filter(d2, msgSentiment==FALSE&replySentiment==FALSE)$mean)
```

```
## [1] 0.5883227
```

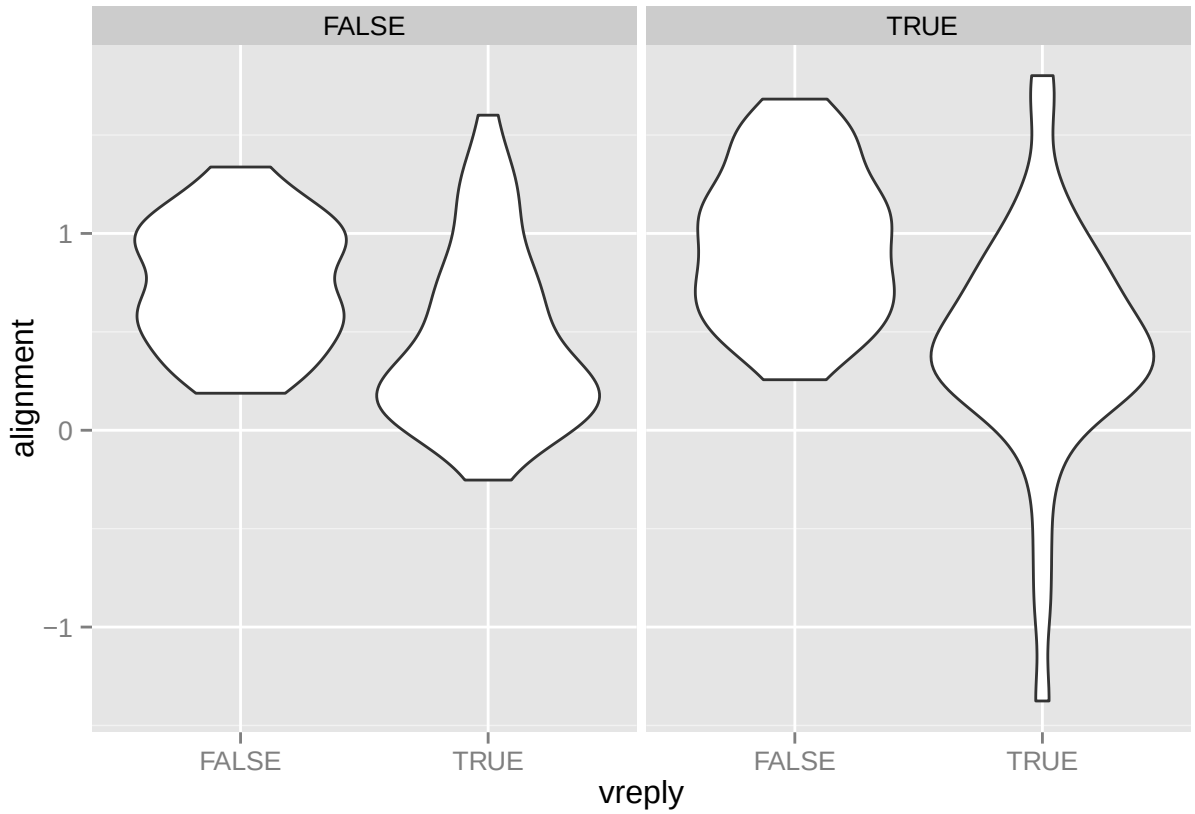
```
nrow(filter(d2, msgSentiment==FALSE&replySentiment==FALSE))
```

```
## [1] 7
```

Ok, so maybe it's an aggregation issue?

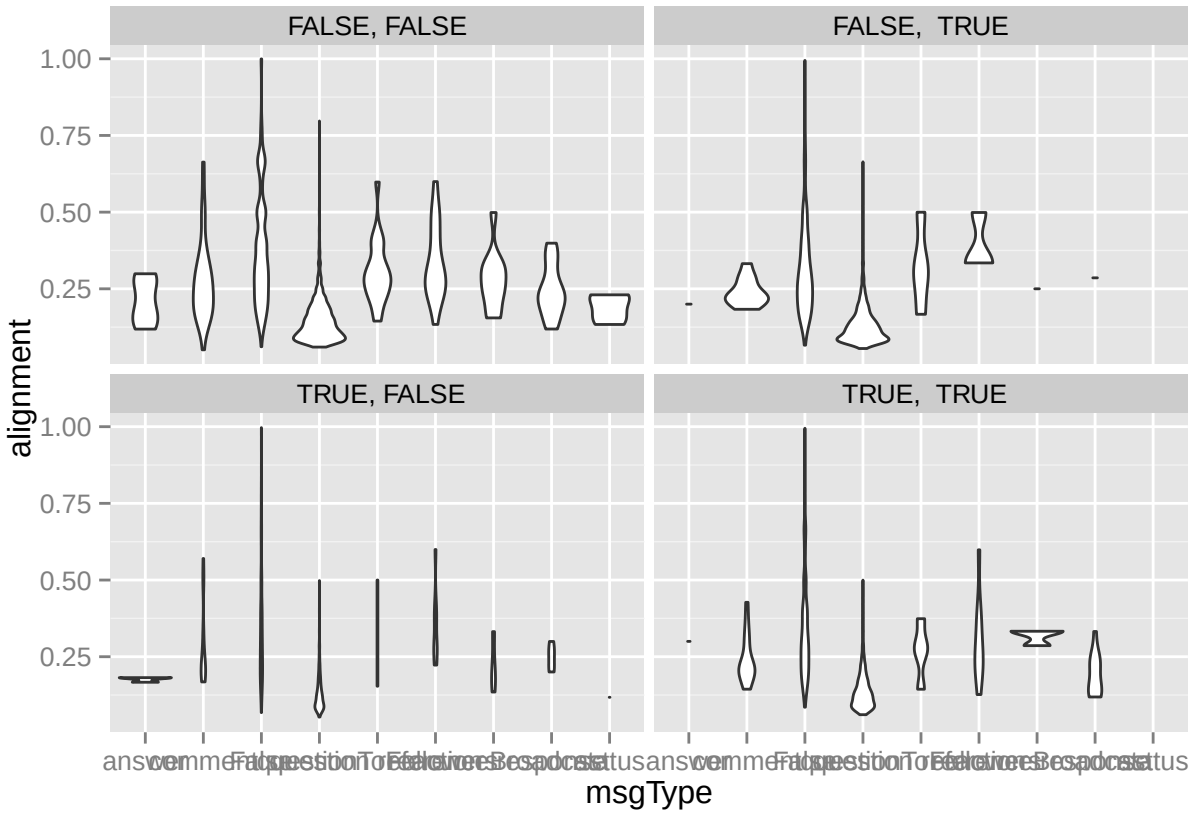
*** Shuffling ***

```
df <- fread('results.csv', header=T)

# Cleaning doun the data table:
d <- df[,list(ba=ba,nba=nba,bna=bna,nbna=nbna,
              vspeak=verifiedSpeaker,vreply=verifiedReplier,
              sid=speakerId,rid=replierId,category=category,
              pyalign=alignment,numUtterances=numUtterances),]

smoothalign <- function(df,sm,align="logodds") {
  if (align=="logodds") {
    return(log(df$ba+sm)-log(df$ba+df$nba+2*sm)-log(df$bna+sm)+log(df$bna+df$nbna+2*sm))
  } else if (align=="subodds") {
    return((df$ba+sm)/(df$ba+df$nba+2*sm)-(df$bna+sm)/(df$bna+df$nbna+2*sm))
  } else if (align=="logdnm") {
    return(log(df$ba+sm)-log(df$ba+df$nba+2*sm)-log(df$bna+df$ba+sm)+log(df$nba+df$ba+df$bna+df$nbna+2*s
  } else if (align=="subdnm") {
    return((df$ba+sm)/(df$ba+df$nba+2*sm)-(df$bna+df$ba+sm)/(df$nba+df$ba+df$bna+df$nbna+2*sm))
  } else {
    stop("Invalid alignment type.")
  }
}

d$lo1 <- smoothalign(d,1,"logodds")

d2 <- d %>%
  filter((ba+nba)>5&(bna+nbna)>5) %>%
  group_by(vspeak, vreply, category) %>%
  summarize(convs=n(),alignment=mean(lo1))

ggplot(d2, aes(x=vreply, y=alignment)) + geom_violin() + facet_wrap(~vspeak)
```



*** Discourse Categories ***

```
df <- fread('discourse_results.csv', header=T)
d <- df %>%
  filter(alignment > 0)

ggplot(d, aes(x=msgType, y=alignment)) + geom_violin() + facet_wrap(verifiedSpeaker~verifiedReplier)
```


FALSE, FALSE
FALSE, TRUE
TRUE, FALSE
TRUE, TRUE
alignment
1.00
0.75
0.50
0.25
msgType