Analysis of Vettius Valens

Definitions

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(cowplot)
```

```
##
## ********************************************************

## Note: As of version 1.0.0, cowplot does not change the

##   default ggplot2 theme anymore. To recover the previous

##   behavior, execute:
##   theme_set(theme_cowplot())

## ********************************************************
```

```
library(tidyr)
library(broman)
```

```
# Order of bodies
ORDER_OF_BODIES = c('Saturn',
                    'Jupiter',
                    'Mars',
                    'Sun',
                    'Venus',
                    'Mercury',
                    'Moon')
names(ORDER_OF_BODIES) <- c("Sa", "J", "Mar", "Su", "V", "Mer", "Moo")
# For plots
theme_basic <- function () {
  theme_bw(base_size=12) %+replace%
    theme(
      axis.text=element_text(colour="black")
    ) %+replace%
    theme(
      panel.grid=element_blank()
    )
}
```

Singles

We read in the sentiments of the singles.

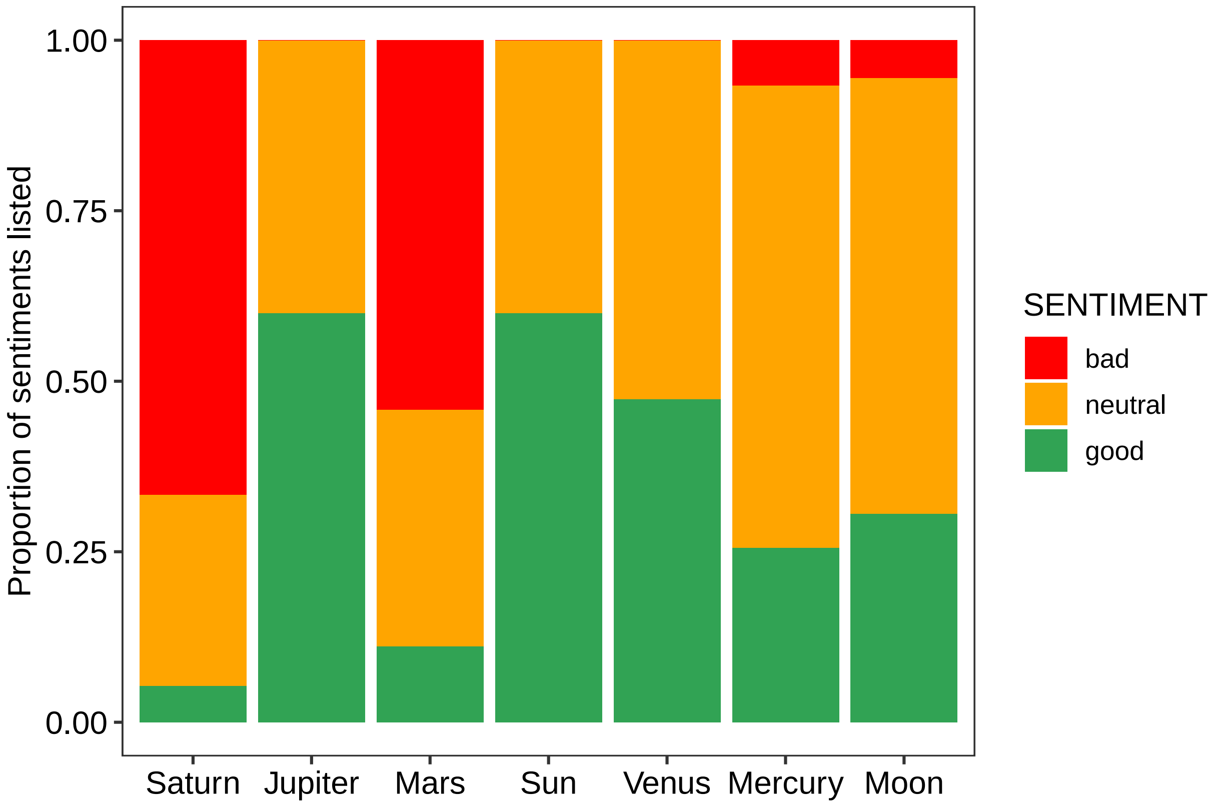
```
singles.sentiments <- read.csv('../../data/singles-qualities.csv',
                               header=T,
                               stringsAsFactors = F)
# Get in useful format
singles.sentiments.df <- singles.sentiments %>% group_by(PLANET, SENTIMENT) %>%
  summarise(count=n()) %>%
  mutate(total=sum(count),
         prop=count/total)
# remove 'total' variable
singles.sentiments.df$total <- NULL
# Make sure list is complete (some sentiments are missing where none listed in input data)
full_data <- expand.grid(PLANET=singles.sentiments.df$PLANET, SENTIMENT=singles.sentiments.df$SENTIMENT)
singles.sentiments.df <- unique(left_join(tbl_df(full_data),singles.sentiments.df))
```

```
## Joining, by = c("PLANET", "SENTIMENT")

## Warning: Column `PLANET` joining factor and character vector, coercing into
## character vector

## Warning: Column `SENTIMENT` joining factor and character vector, coercing into
## character vector
```

```
singles.sentiments.df$count[is.na(singles.sentiments.df$count)] <- 0
singles.sentiments.df$prop[is.na(singles.sentiments.df$prop)] <- 0
# Order planets and sentiments
singles.sentiments.df$PLANET <- ordered(singles.sentiments.df$PLANET,
                                        levels=ORDER_OF_BODIES)
singles.sentiments.df$SENTIMENT <- ordered(singles.sentiments.df$SENTIMENT,
                                         levels=c("bad", "neutral", "good"))
# Overall sentiment index
singles.overall.sentiments <- singles.sentiments.df %>% group_by(PLANET) %>%
  summarise(sentiment=prop[which(SENTIMENT=="good")]-prop[which(SENTIMENT=="bad")])
```

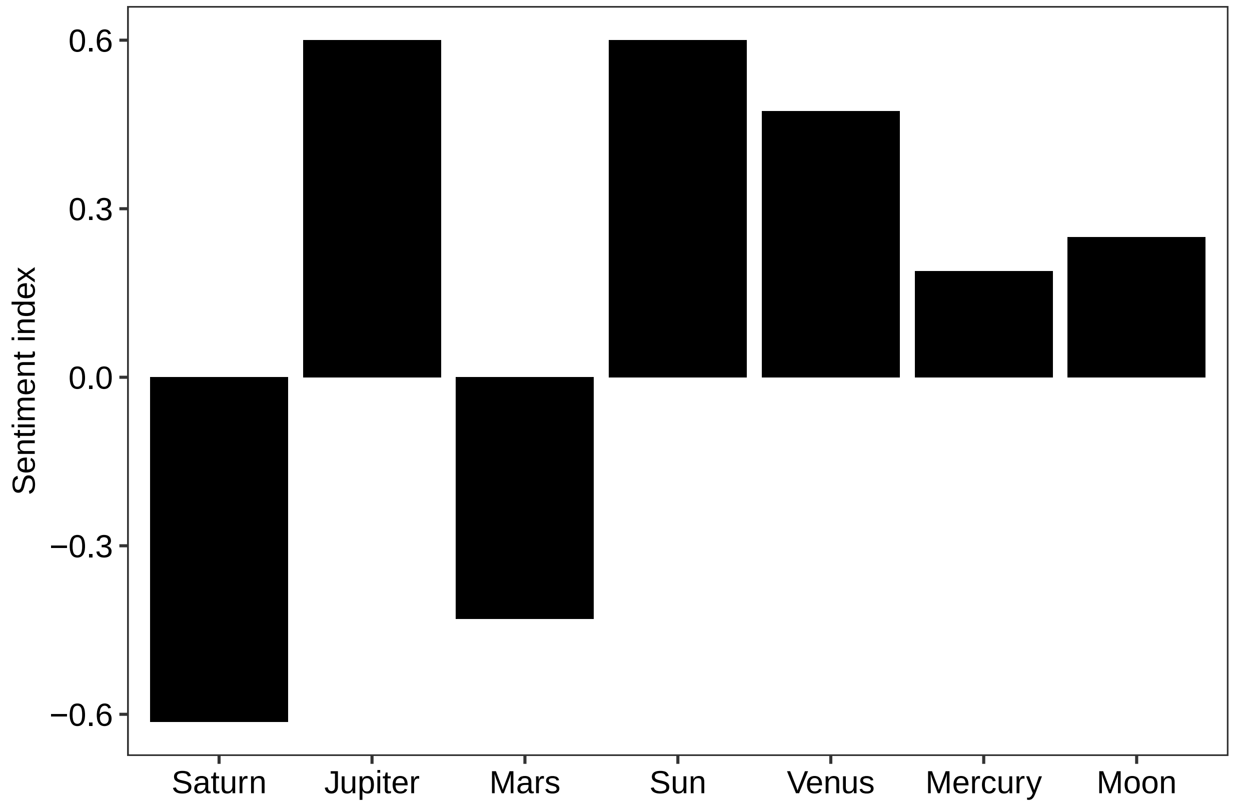
We then plot the sentiments of the singles.

```
# Plot them
ggplot(singles.sentiments.df, aes(PLANET, prop, fill=SENTIMENT))+
  geom_bar(stat="identity")+
  theme_basic()+
  xlab("")+
  scale_fill_manual(values=c("red", "orange", "#31a354"))+
  ylab("Proportion of sentiments listed")
```



Also plot overall sentiment index (good - bad).

```
ggplot(singles.overall.sentiments, aes(PLANET, sentiment))+
  geom_bar(stat="identity", fill="black")+
  theme_basic()+
  xlab("")+
  ylab("Sentiment index")
```



# Doubles

We read in the sentiments of the double conjunctions (e.g. Saturn and Jupiter).

```
doubles.sentiments <- read.csv('../../data/doubles-qualities.csv',
                               header=T,
                               stringsAsFactors = F)
# Get in useful format
doubles.sentiments.df <- doubles.sentiments %>% group_by(DOUBLE, SENTIMENT) %>%
  summarise(count=n()) %>%
  mutate(total=sum(count),
         prop=count/total)
# remove 'total' variable
doubles.sentiments.df$total <- NULL
# Make sure list is complete (some sentiments are missing where none listed in input data)
full_data <- expand.grid(DOUBLE=doubles.sentiments.df$DOUBLE, SENTIMENT=doubles.sentiments.df$SENTIMENT
doubles.sentiments.df <- unique(left_join(tbl_df(full_data),doubles.sentiments.df))
```

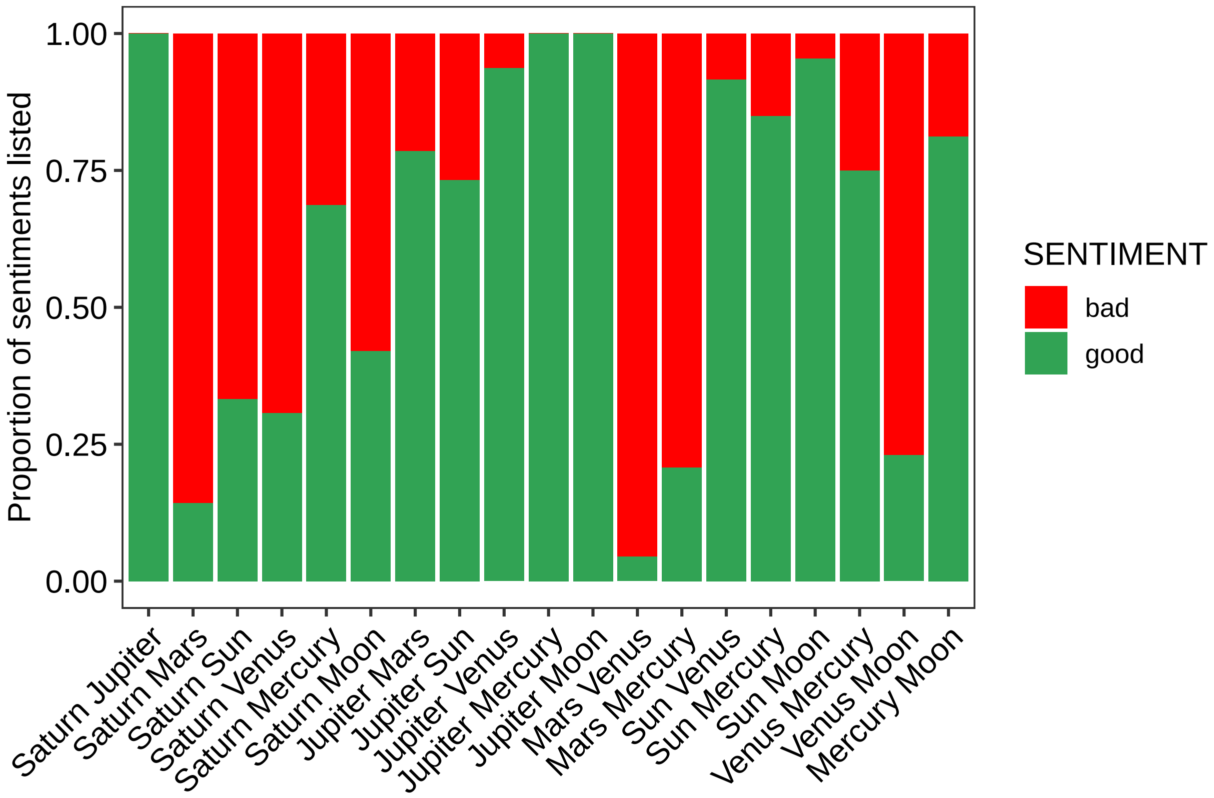
```
## Joining, by = c("DOUBLE", "SENTIMENT")

## Warning: Column `DOUBLE` joining factor and character vector, coercing into
## character vector

## Warning: Column `SENTIMENT` joining factor and character vector, coercing into
## character vector
```

```
doubles.sentiments.df$count[is.na(doubles.sentiments.df$count)] <- 0
doubles.sentiments.df$prop[is.na(doubles.sentiments.df$prop)] <- 0



# Order planets and sentiments
```

```
doubles.sentiments.df$body.1 <- sapply(stringr::str_split(doubles.sentiments.df$DOUBLE, pattern=" "),
                              function(x) x[1])
doubles.sentiments.df$body.1 <- ordered(ORDER_OF_BODIES[doubles.sentiments.df$body.1],
                               levels=ORDER_OF_BODIES)
doubles.sentiments.df$body.2 <- sapply(stringr::str_split(doubles.sentiments.df$DOUBLE, pattern=" "),
                              function(x) x[2])
doubles.sentiments.df$body.2 <- ordered(ORDER_OF_BODIES[doubles.sentiments.df$body.2],
                               levels=ORDER_OF_BODIES)
doubles.sentiments.df$bodies.sorted <- sapply(1:nrow(doubles.sentiments.df),
                                     function(x)
                                       paste(as.character(sort(unlist(c(doubles.sentiments.df[x,"body.1
                                                                        doubles.sentiments.df[x,"body.2
                                             collapse=" "))
# Make body 1 and 2 be in order of bodies as expected
doubles.sentiments.df$body.1 <- gsub(" .*", "", doubles.sentiments.df$bodies.sorted)
doubles.sentiments.df$body.2 <- gsub(".* ", "", doubles.sentiments.df$bodies.sorted)
doubles.sentiments.df$body.1 <- ordered(doubles.sentiments.df$body.1,
                                       levels=ORDER_OF_BODIES)
doubles.sentiments.df$body.2 <- ordered(doubles.sentiments.df$body.2,
                                       levels=ORDER_OF_BODIES)
doubles.sentiments.df$order.body.string <- paste0(as.numeric(doubles.sentiments.df$body.1),
                                                as.numeric(doubles.sentiments.df$body.2))
# Order bodies again (this is hacky but it works)
doubles.sentiments.df$bodies.sorted <- ordered(doubles.sentiments.df$bodies.sorted,
                                              levels=unique(doubles.sentiments.df$bodies.sorted[order(d

doubles.sentiments.df$SENTIMENT <- ordered(doubles.sentiments.df$SENTIMENT,
                                        levels=c("bad", "good"))

# Also make a dataframe with an overall sentiment index
doubles.overall.sentiments <- doubles.sentiments.df %>% group_by(body.1, body.2, bodies.sorted, order.b
  summarise(sentiment=prop[which(SENTIMENT=="good")]-prop[which(SENTIMENT=="bad")])
```

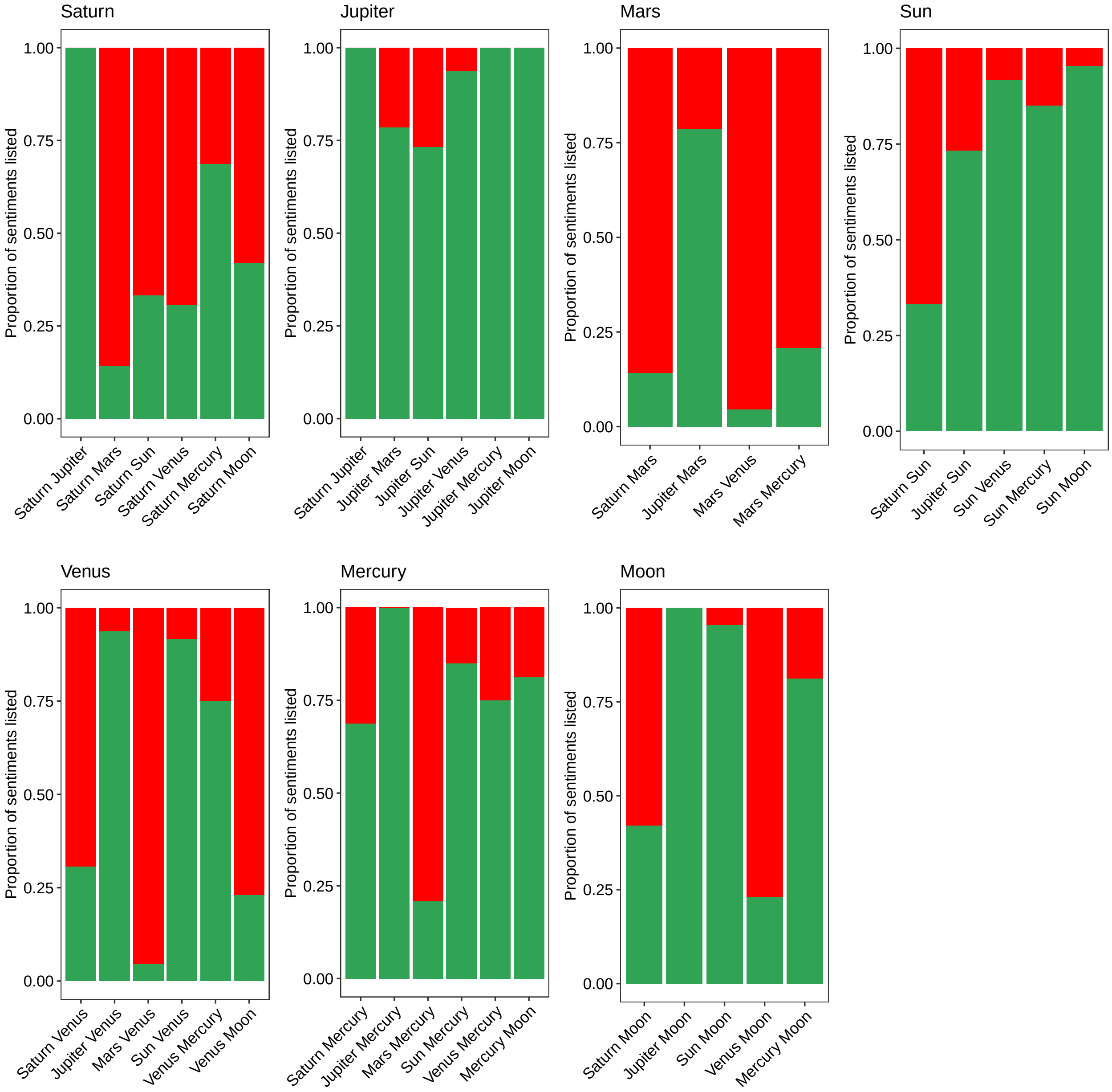
We then plot the sentiments of the doubles.

```
# Plot them
ggplot(doubles.sentiments.df, aes(bodies.sorted, prop, fill=SENTIMENT))+
  geom_bar(stat="identity")+
  theme_basic()+
  xlab("")+
  scale_fill_manual(values=c("red",  "#31a354"))+
  ylab("Proportion of sentiments listed")+
  theme(axis.text.x=element_text(angle=45, hjust=1))
```



A function to plot all the doubles involving a particular planet.

```
plotDoublesWithPlanet <- function(planet){
  # Subset to only doubles involving planet
  local.planet.df <- doubles.sentiments.df[grep(planet, doubles.sentiments.df$bodies.sorted),]
  ggplot(local.planet.df, aes(bodies.sorted, prop, fill=SENTIMENT))+
  geom_bar(stat="identity")+
  theme_basic()+
  xlab("")+
  scale_fill_manual(values=c("red", "#31a354"))+
  ylab("Proportion of sentiments listed")+
  theme(axis.text.x=element_text(angle=45, hjust=1))+
    ggtitle(planet)+
    theme(legend.position = "none")
}
# Make all these plots and combine them
p.doubles.saturn <- plotDoublesWithPlanet("Saturn")
p.doubles.jupiter <- plotDoublesWithPlanet("Jupiter")
p.doubles.mars <- plotDoublesWithPlanet("Mars")
p.doubles.sun <- plotDoublesWithPlanet("Sun")
p.doubles.venus <- plotDoublesWithPlanet("Venus")
p.doubles.mercury <- plotDoublesWithPlanet("Mercury")
p.doubles.moon <- plotDoublesWithPlanet("Moon")
cowplot::plot_grid(p.doubles.saturn, p.doubles.jupiter,
                   p.doubles.mars, p.doubles.sun,
                   p.doubles.venus, p.doubles.mercury,
                   p.doubles.moon, nrow=2)
```



Note that Valens misses out two doubles which involve Mars: Mars + Sun, and Mars + Moon.

Predicting doubles from singles

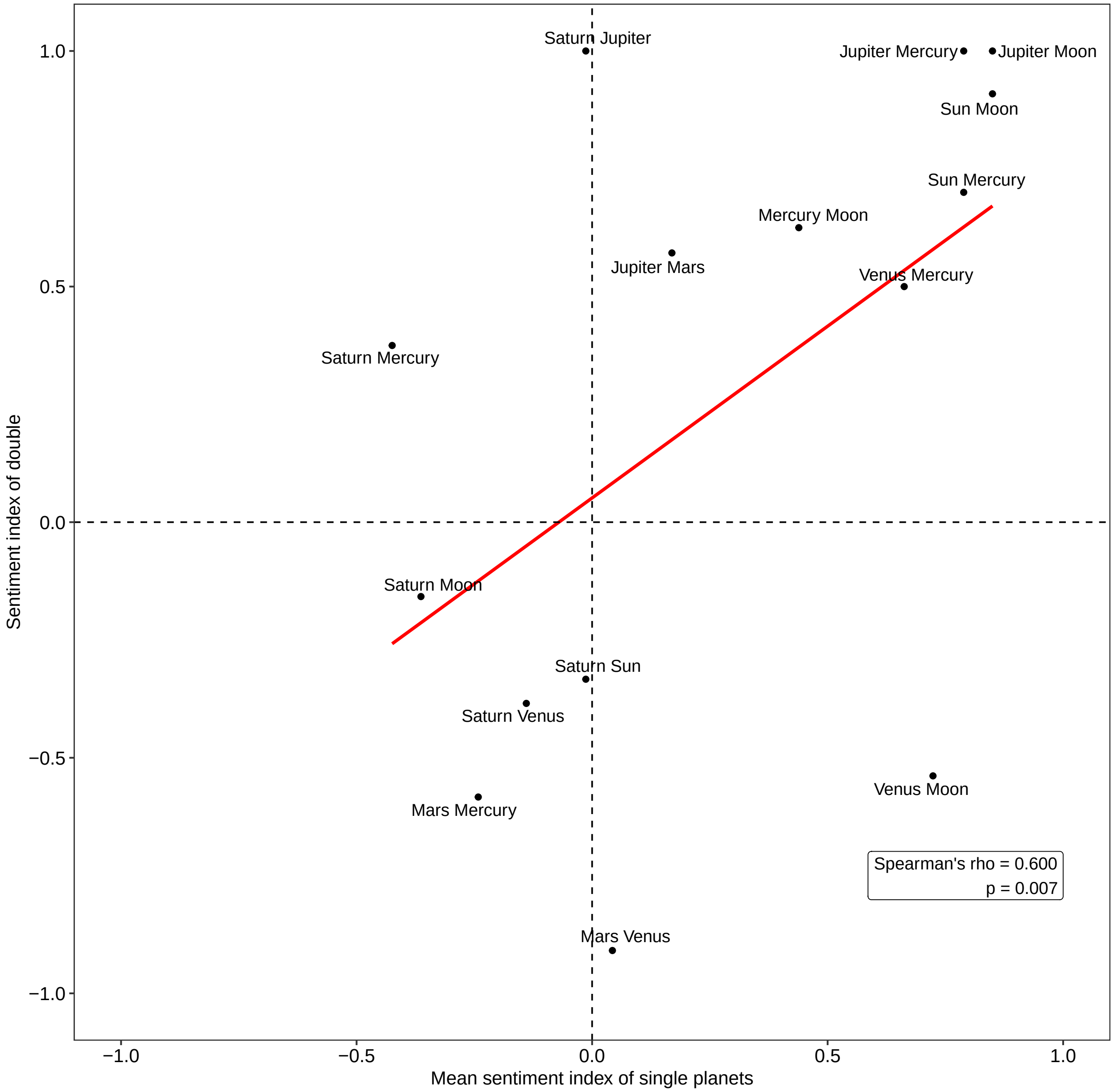
If we know the sentiment proportions associated with single planets, can we predict the sentiment of the doubles?

```
# getSingleScores <- function(double, sentiment="good", mean=TRUE, singles=singles.sentiments.df){
#     double.bodies <- ordered(unlist(stringr::str_split(double, pattern=" ")),
#                             levels=ORDER_OF_BODIES)
#     prop <- 0
#     for (body in double.bodies){
#       new.prop <- as.numeric(singles[which(singles$PLANET==body &
#                                       singles$SENTIMENT==sentiment), "prop"]  )
#       prop <- prop +  new.prop
#     }
#     if (mean==FALSE){
#       return(prop)
#     }
#     else{
#       return(prop/2)
#     }
# }
getSingleScores <- function(conjunction, mean=TRUE, singles=singles.overall.sentiments){
    conjunction.bodies <- ordered(unlist(stringr::str_split(conjunction, pattern=" ")),
                           levels=ORDER_OF_BODIES)
    prop <- 0
    for (body in conjunction.bodies){
      new.prop <- as.numeric(singles[which(singles$PLANET==body), "sentiment"])
      prop <- prop +  new.prop
    }
    if (mean==FALSE){
      return(prop)
    }
    else{
      return(prop/length(prop))
    }
}

doubles.overall.sentiments$mean.single.sentiments <- sapply(doubles.overall.sentiments$bodies.sorted,
                                                          function(x) getSingleScores(x, mean=TRUE))
# Add correlation score
spearman.cor <- cor.test(doubles.overall.sentiments$mean.single.sentiments, doubles.overall.sentiments$s
```

```
## Warning in cor.test.default(doubles.overall.sentiments$mean.single.sentiments, :
## Cannot compute exact p-value with ties
```

```
p.sentiments.doubles.singles <- ggplot(doubles.overall.sentiments, aes(mean.single.sentiments, sentiment
    stat_smooth(method="lm", se=FALSE, colour="red")+
  geom_point(colour="black")+
  theme_basic()+
  ggrepel::geom_text_repel(aes(label=bodies.sorted))+
    coord_fixed()+
  xlim(c(-1, 1))+
  ylim(c(-1,1))+
  geom_hline(yintercept = 0, linetype='dashed')+
    geom_vline(xintercept = 0, linetype='dashed')+
  xlab("Mean sentiment index of single planets")+
  ylab("Sentiment index of double")+
  annotate(geom="label", label=paste0("Spearman's rho = ", myround(spearman.cor$estimate, 3), "\n",
                                     "p = ", myround(spearman.cor$p.value, 3)), x=1, y=-0.75, hjust=1)
p.sentiments.doubles.singles
```

```
## Warning: Removed 4 rows containing non-finite values (stat_smooth).

## Warning: Removed 4 rows containing missing values (geom_point).

## Warning: Removed 4 rows containing missing values (geom_text_repel).
```



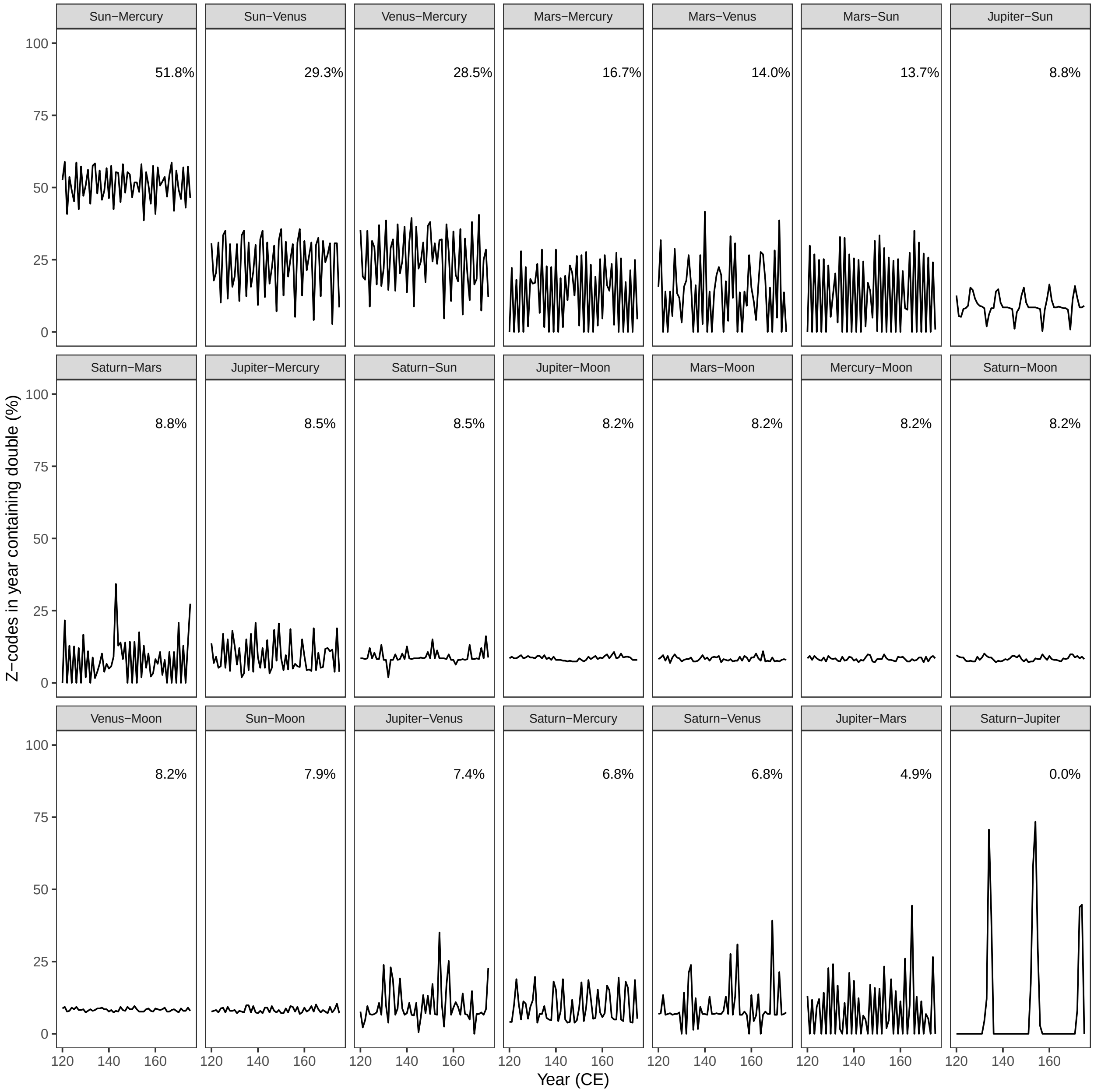
We see that there is a strong correlation between the mean sentiment index for the component singles and the sentiment index of the double conjunction.

Double conjuction occurrences

Read in the frequencies of the double conjunctions.

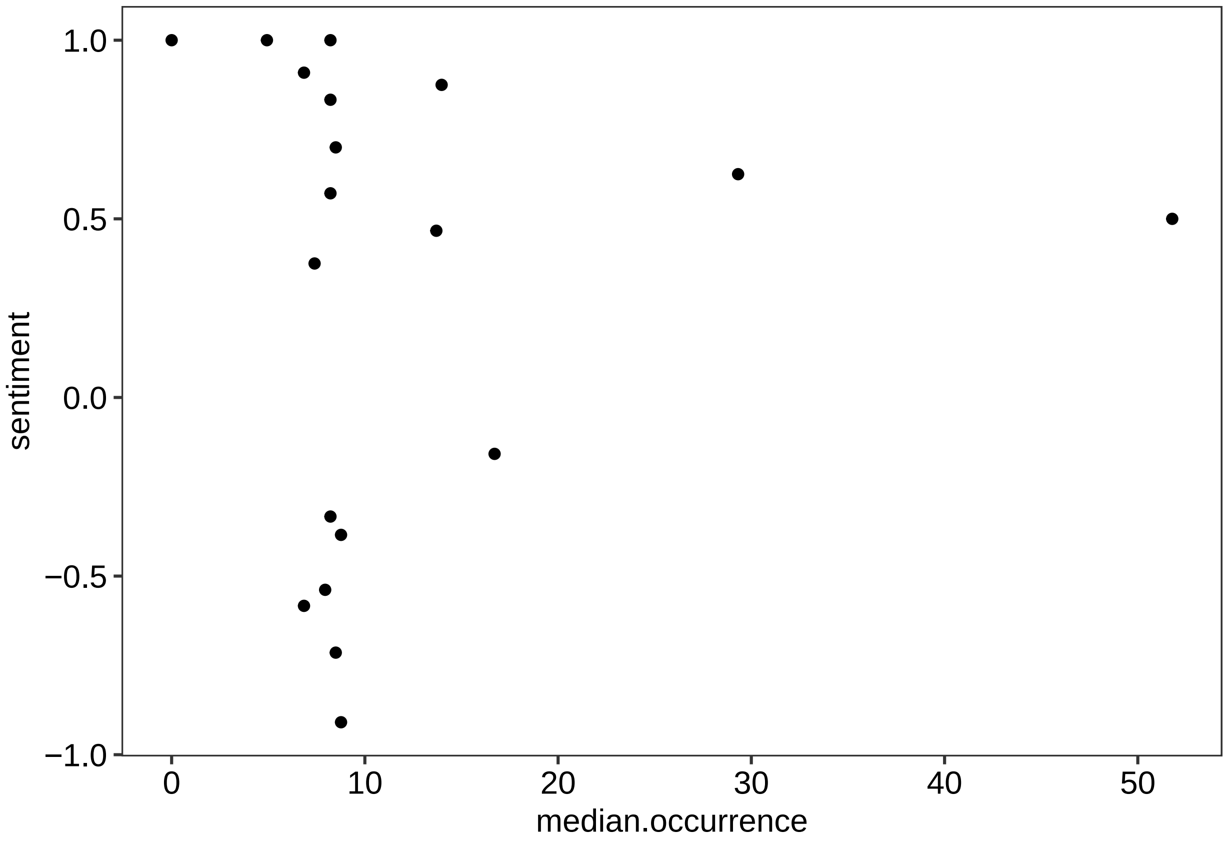
```
double.occurrences <- read.csv('../../data/0-CE-200-CE-double-occurrence-per-year.csv', header=F,string
colnames(double.occurrences) <- c("year", "double", "occurrence")
double.occurrences.median <- double.occurrences %>% group_by(double) %>%
  summarise(median=median(occurrence))
order.of.doubles <- double.occurrences.median$double[order(double.occurrences.median$median, decreasing
double.occurrences.median$year <- 160
double.occurrences.median$y <- 11
double.occurrences.median$double <- ordered(double.occurrences.median$double,
                                     levels=order.of.doubles)
double.occurrences.median$median.plot <- myround(double.occurrences.median$median, 1)
double.occurrences$double <- ordered(double.occurrences$double,
                              levels=order.of.doubles)

# Make plot
occurrences.plot <- ggplot(double.occurrences, aes(year, occurrence, group=double))+
  geom_line()+
  theme_bw()+
  xlab("Year (CE)")+
  ylab("Z-codes in year containing double (%)")+
  facet_wrap(~double, ncol=7)+
  theme(panel.grid = element_blank())+
  ylim(c(0,100))+
  xlim(c(120, 175))+
  theme(strip.text = element_text(size=8))+
  geom_text(data=double.occurrences.median,
            aes(year, 90, label=paste0(median.plot, "%"), group=double),
            hjust=0, size=3)
occurrences.plot
```

```
## Warning: Removed 3003 rows containing missing values (geom_path).
```



Does median occurrence relate to mean sentiment index?

```
double.occurrences.median$bodies.sorted <- ordered(gsub("-", " ",double.occurrences.median$double),
                                                   levels=levels(doubles.overall.sentiments$bodies.sorte
median.occurrences.doubles <- double.occurrences.median$median
names(median.occurrences.doubles) <- double.occurrences.median$bodies.sorted
# Add to sentiment
doubles.overall.sentiments$median.occurrence <- median.occurrences.doubles[doubles.overall.sentiments$bo
ggplot(doubles.overall.sentiments, aes(median.occurrence, sentiment))+
  geom_point()+
  theme_basic()
```



Not obviously. But perhaps there is a linear model term we can use.

```
summary(lm(sentiment ~ mean.single.sentiments + median.occurrence, data=doubles.overall.sentiments))
```

```
##
## Call:
## lm(formula = sentiment ~ mean.single.sentiments + median.occurrence,
##     data = doubles.overall.sentiments)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.07939 -0.35603  0.05497  0.33136  0.95689
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.0565613  0.1916160   0.295  0.77165
## mean.single.sentiments  0.6790886  0.2083126   3.260  0.00492 **
## median.occurrence      -0.0008909  0.0114380  -0.078  0.93888
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5439 on 16 degrees of freedom
## Multiple R-squared:  0.4017, Adjusted R-squared:  0.327
## F-statistic: 5.372 on 2 and 16 DF,  p-value: 0.01641
```

Again, it appears not.

Triples

```
triples.sentiments <- read.csv('../../data/triples-qualities.csv',
                              header=T,
                              stringsAsFactors = F)
# Get in useful format
triples.sentiments.df <- triples.sentiments %>% group_by(TRIPLE, SENTIMENT) %>%
  summarise(count=n()) %>%
  mutate(total=sum(count),
         prop=count/total)
# remove 'total' variable
triples.sentiments.df$total <- NULL
# Make sure list is complete (some sentiments are missing where none listed in input data)
full_data <- expand.grid(TRIPLE=triples.sentiments.df$TRIPLE, SENTIMENT=triples.sentiments.df$SENTIMENT
triples.sentiments.df <- unique(left_join(tbl_df(full_data),triples.sentiments.df))
```

```
## Joining, by = c("TRIPLE", "SENTIMENT")
```

```
## Warning: Column `TRIPLE` joining factor and character vector, coercing into
## character vector
```

```
## Warning: Column `SENTIMENT` joining factor and character vector, coercing into
## character vector
```

```
triples.sentiments.df$count[is.na(triples.sentiments.df$count)] <- 0
triples.sentiments.df$prop[is.na(triples.sentiments.df$prop)] <- 0



# Order planets and sentiments
triples.sentiments.df$body.1 <- sapply(stringr::str_split(triples.sentiments.df$TRIPLE, pattern=" "),
                              function(x) x[1])
triples.sentiments.df$body.1 <- ordered(ORDER_OF_BODIES[triples.sentiments.df$body.1],
                               levels=ORDER_OF_BODIES)
triples.sentiments.df$body.2 <- sapply(stringr::str_split(triples.sentiments.df$TRIPLE, pattern=" "),
                              function(x) x[2])
triples.sentiments.df$body.2 <- ordered(ORDER_OF_BODIES[triples.sentiments.df$body.2],
                               levels=ORDER_OF_BODIES)
triples.sentiments.df$body.3 <- sapply(stringr::str_split(triples.sentiments.df$TRIPLE, pattern=" "),
                              function(x) x[3])
triples.sentiments.df$body.3 <- ordered(ORDER_OF_BODIES[triples.sentiments.df$body.3],
                               levels=ORDER_OF_BODIES)
triples.sentiments.df$bodies.sorted <- sapply(1:nrow(triples.sentiments.df),
                                     function(x)
                                       paste(as.character(sort(unlist(c(triples.sentiments.df[x,"body.1
                                                                        triples.sentiments.df[x,"body.2
                                             collapse=" "))
# Make body 1 and 2 be in order of bodies as expected
triples.sentiments.df$body.1 <- sapply(stringr::str_split(triples.sentiments.df$bodies.sorted, pattern=
                              function(x) x[1])
triples.sentiments.df$body.2 <- sapply(stringr::str_split(triples.sentiments.df$bodies.sorted, pattern=
                              function(x) x[2])
triples.sentiments.df$body.3 <- sapply(stringr::str_split(triples.sentiments.df$bodies.sorted, pattern=
                              function(x) x[3])
triples.sentiments.df$body.1 <- ordered(triples.sentiments.df$body.1,
                                       levels=ORDER_OF_BODIES)
```

```
triples.sentiments.df$body.2 <- ordered(triples.sentiments.df$body.2,
                                       levels=ORDER_OF_BODIES)
triples.sentiments.df$body.3 <- ordered(triples.sentiments.df$body.3,
                                       levels=ORDER_OF_BODIES)

triples.sentiments.df$order.body.string <- paste0(as.numeric(triples.sentiments.df$body.1),
                                                as.numeric(triples.sentiments.df$body.2),
                                                as.numeric(triples.sentiments.df$body.3))
# Order bodies again (this is hacky but it works)
triples.sentiments.df$bodies.sorted <- ordered(triples.sentiments.df$bodies.sorted,
                                              levels=unique(triples.sentiments.df$bodies.sorted[order(t

triples.sentiments.df$SENTIMENT <- ordered(triples.sentiments.df$SENTIMENT,
                                        levels=c("bad", "neutral", "good"))

# Also make a dataframe with an overall sentiment index
triples.overall.sentiments <- triples.sentiments.df %>% group_by(body.1, body.2,body.3, bodies.sorted, 
  summarise(sentiment=prop[which(SENTIMENT=="good")]-prop[which(SENTIMENT=="bad")])
```

Now we see how this correlates with component singles or doubles.

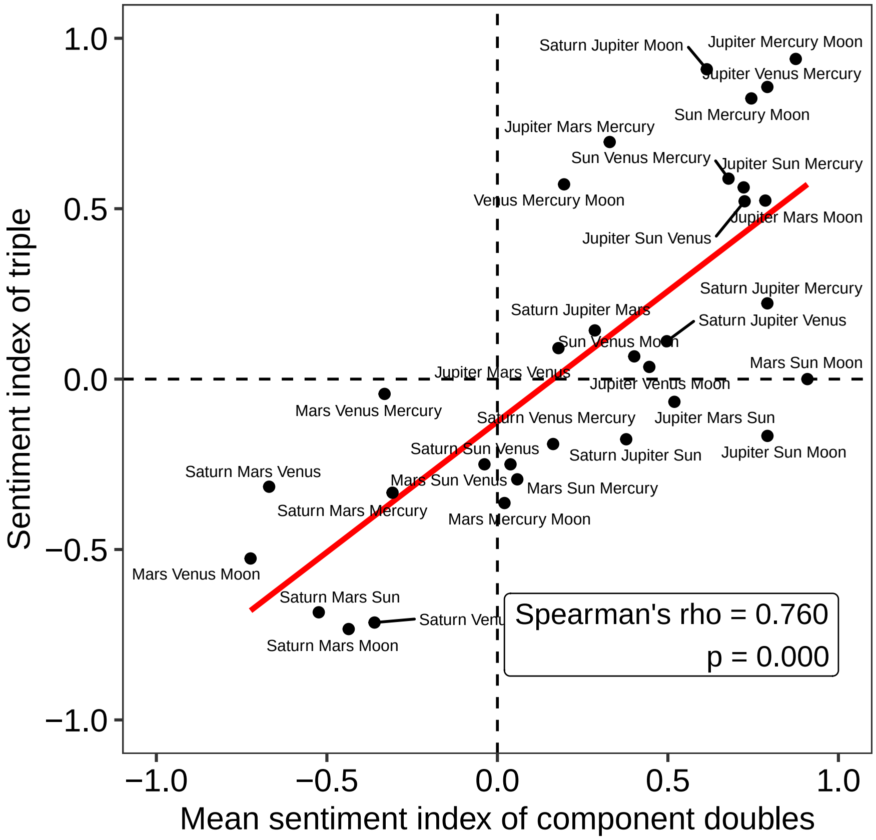
```
getDoubleScoresForTriple <- function(triple, double.data=doubles.overall.sentiments, mean=TRUE){
  triple.bodies <- ordered(unlist(stringr::str_split(triple, pattern=" ")),
                           levels=ORDER_OF_BODIES)

  # combinations
  doubles <- ordered(combn(triple.bodies, 2), levels=ORDER_OF_BODIES)
  doubles <- sapply(c(1, 3, 5), function(x) paste(c(doubles[x], doubles[x+1]), collapse= ' '))

  doubles.ordered <- sapply( doubles,
                   function(x) paste(as.character(sort(unlist(strsplit(fixed = TRUE, split=" ", x))), c
  doubles.ordered <- as.data.frame(doubles.ordered)
  set.of.doubles <- sapply(c(1,2, 3),
         function(x) paste(ORDER_OF_BODIES[as.numeric(as.character(doubles.ordered[1,x]))],
                           ORDER_OF_BODIES[as.numeric(as.character(doubles.ordered[2,x]))]))

  prop <- 0
  i <- 0
  props <- c()
  for (double in set.of.doubles){
    if (!double  %in% c("Mars Sun", "Mars Moon")){
      new.prop <- double.data[which(double.data$bodies.sorted==double),"sentiment"]
      prop <- prop +  new.prop
      props <- c(props, as.numeric(new.prop))
      i <- i+1
    }

  }
  mean.prop <- prop/i
  if (mean==TRUE){
    return(as.numeric(mean.prop))
  }
  else{
    return(sum(props))
  }
}
triples.overall.sentiments$mean.double.sentiment <- sapply(triples.overall.sentiments$bodies.sorted, fu
```

Now we plot this relationship.

```
spearman.cor.triple.double <- cor.test(triples.overall.sentiments$mean.double.sentiment, triples.overal
```

```
## Warning in cor.test.default(triples.overall.sentiments$mean.double.sentiment, :
## Cannot compute exact p-value with ties
```

```
p.sentiments.triples.doubles <- ggplot(triples.overall.sentiments, aes(mean.double.sentiment, sentiment
    stat_smooth(method="lm", se=FALSE, colour="red")+
  geom_point(colour="black")+
  theme_basic()+
  ggrepel::geom_text_repel(aes(label=bodies.sorted), size=2)+
    coord_fixed()+
  xlim(c(-1, 1))+
  ylim(c(-1,1))+
  geom_hline(yintercept = 0, linetype='dashed')+
    geom_vline(xintercept = 0, linetype='dashed')+
  xlab("Mean sentiment index of component doubles")+
  ylab("Sentiment index of triple")+
  annotate(geom="label", label=paste0("Spearman's rho = ", myround(spearman.cor.triple.double$estimate,
                                     "p = ", myround(spearman.cor.triple.double$p.value, 3)), x=1, y=-0
p.sentiments.triples.doubles
```



Is this stronger or weaker if we just consider component singles?

```
triples.overall.sentiments$mean.single.sentiment <- sapply(triples.overall.sentiments$bodies.sorted, fu
spearman.cor.triple.single <- cor.test(triples.overall.sentiments$mean.single.sentiment, triples.overal
```

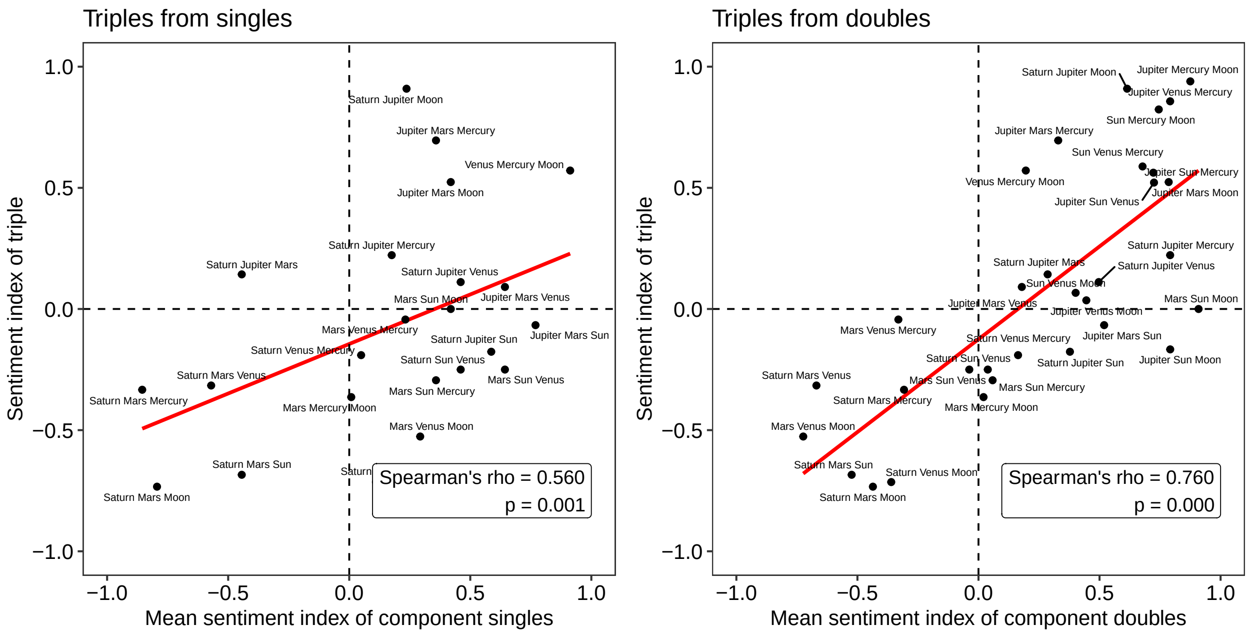
```
## Warning in cor.test.default(triples.overall.sentiments$mean.single.sentiment, :
## Cannot compute exact p-value with ties
```

```
p.sentiments.triples.singles <- ggplot(triples.overall.sentiments, aes(mean.single.sentiment, sentiment
    stat_smooth(method="lm", se=FALSE, colour="red")+
  geom_point(colour="black")+
  theme_basic()+
  ggrepel::geom_text_repel(aes(label=bodies.sorted), size=2)+
    coord_fixed()+
  xlim(c(-1, 1))+
  ylim(c(-1,1))+
  geom_hline(yintercept = 0, linetype='dashed')+
    geom_vline(xintercept = 0, linetype='dashed')+
  xlab("Mean sentiment index of component singles")+
  ylab("Sentiment index of triple")+
  annotate(geom="label", label=paste0("Spearman's rho = ", myround(spearman.cor.triple.single$estimate,
                                     "p = ", myround(spearman.cor.triple.single$p.value, 3)), x=1, y=-0
# Plot both of them
cowplot::plot_grid(p.sentiments.triples.singles+ggtitle("Triples from singles"), p.sentiments.triples.d
```

```
## Warning: Removed 9 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```

```
## Warning: Removed 9 rows containing missing values (geom_text_repel).
```



Triples from singles

Triples from doubles