R Notebook

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
##
## 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(ggplot2)
library(urca)
```

```
## Warning: package 'urca' was built under R version 4.0.2
```

```
library(TTR)
```

```
## Warning: package 'TTR' was built under R version 4.0.2
```

```
library(pastecs)
```

```
## Warning: package 'pastecs' was built under R version 4.0.2
```

```
##
## Attaching package: 'pastecs'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
library(TSclust)
```

```
## Warning: package 'TSclust' was built under R version 4.0.2
```

```
## Loading required package: pdc
```

```
## Warning: package 'pdc' was built under R version 4.0.2
```

```
## Loading required package: cluster
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.0.2
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:pastecs':
##
##     extract
```

```
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.0.2
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

Load data and add genres

```
ratings<-read.csv("ratings.csv")
movie_genres<-read.csv("movie_genres.csv")
ratings_genres<-ratings %>% select(movieId,rating,timestamp) %>% left_join( movie_genres %>% select(movi
```

Kruskal-Wallis H Test

```
kruskal.test(rating~genres,data=ratings_genres)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  rating by genres
## Kruskal-Wallis chi-squared = 3306.4, df = 19, p-value < 2.2e-16
```

Conclusion: p-value < 0.05, grouping by genres makes sense.

Calculate the age of movie when being rated (for a given movie, age:=timestamp-min(timestamp))

```
movie_firsttime<-summarise(group_by(ratings_genres,movieId),firsttime=min(timestamp))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ratings_genres_age<-ratings_genres %>% left_join(movie_firsttime, by= c("movieId"="movieId"))
ratings_genres_age<-ratings_genres_age %>% mutate(age=timestamp-firsttime) %>% select(rating,genres,age
```

Normalize the ratings based on genres (for all age together)

```
ratings_genres_age_normalize_group<-group_by(ratings_genres_age,genres)
ratings_genres_age_normalize<-summarise(ratings_genres_age_normalize_group,rating_normalize=scale(ratin
```

```
## `summarise()` regrouping output by 'genres' (override with `.groups` argument)
```

Calculate the mean rating for each age based on genres (without normalization)

```
ratings_genres_age_cut<-ratings_genres_age
ratings_genres_age_cut$age<-cut(ratings_genres_age_cut$age,100)
```

```
ratings_genres_age_cut_group<-group_by(ratings_genres_age_cut,genres,age)
ratings_genres_age_ts<-summarise(ratings_genres_age_cut_group,mean_rating=mean(rating))
```

```
## `summarise()` regrouping output by 'genres' (override with `.groups` argument)
```

```
ratings_genres_age_ts<-filter(ratings_genres_age_ts,genres!="(no genres listed)")
```

Calculate the mean rating for each age based on genres (with normalization)

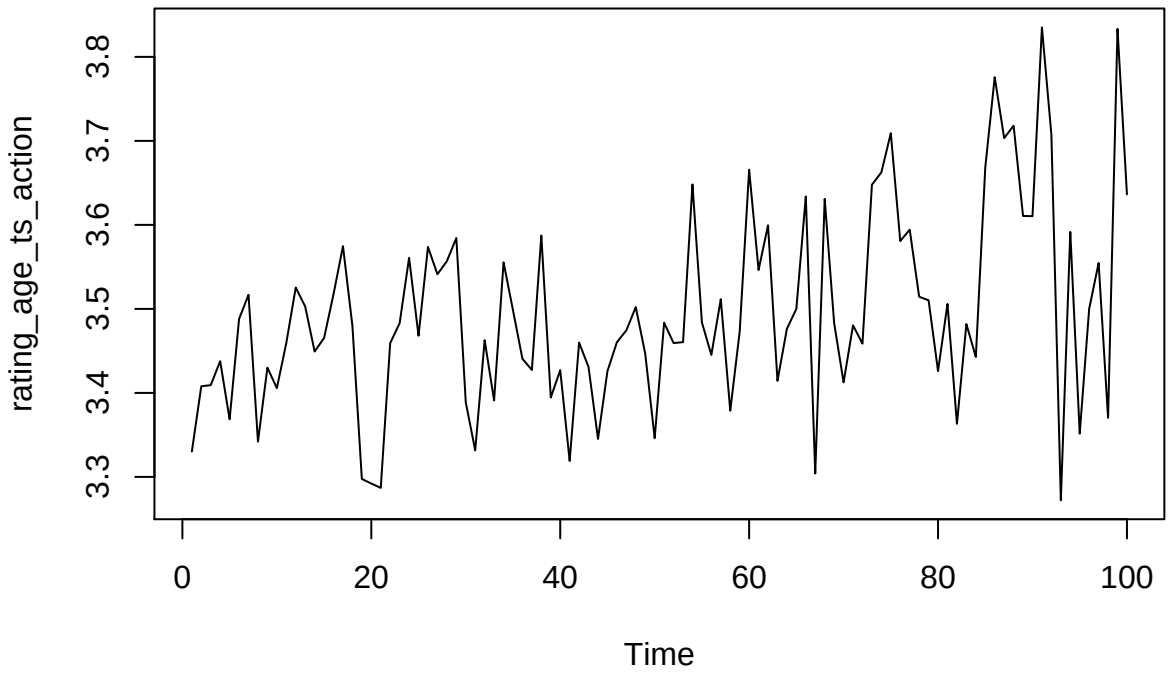
```
ratings_genres_age_normalize_cut<-ratings_genres_age_normalize
ratings_genres_age_normalize_cut$age<-cut(ratings_genres_age_normalize_cut$age,100)
ratings_genres_age_normalize_cut_group<-group_by(ratings_genres_age_normalize_cut,genres,age)
ratings_genres_age_normalize_ts<-summarise(ratings_genres_age_normalize_cut_group,mean_rating_normalize=
```

```
## `summarise()` regrouping output by 'genres' (override with `.groups` argument)
```

Take Action movie for example

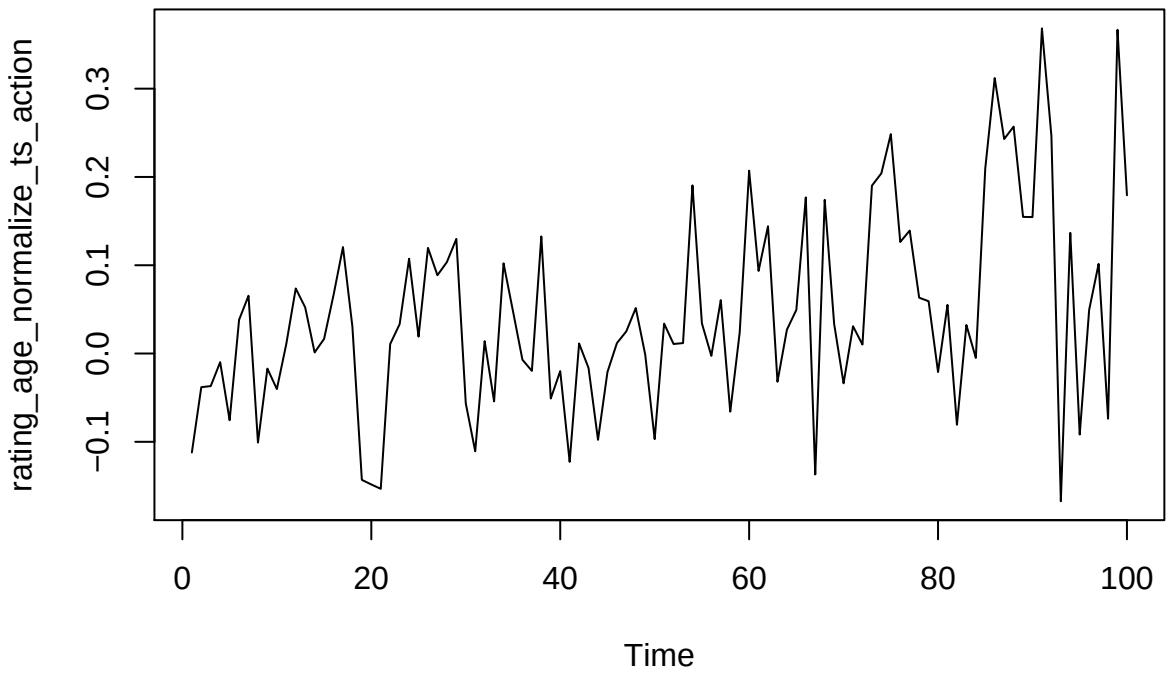
Plot (without normalization)

```
rating_age_ts_action<-subset(ratings_genres_age_ts,genres=="Action")
rating_age_ts_action<-ts(rating_age_ts_action$mean_rating)
plot(rating_age_ts_action)
```



Plot (with normalization)

```
rating_age_normalize_ts_action<-subset(ratings_genres_age_normalize_ts,genres=="Action")
rating_age_normalize_ts_action<-ts(rating_age_normalize_ts_action$mean_rating_normalize)
plot(rating_age_normalize_ts_action)
```



Stationarity test (kpss)

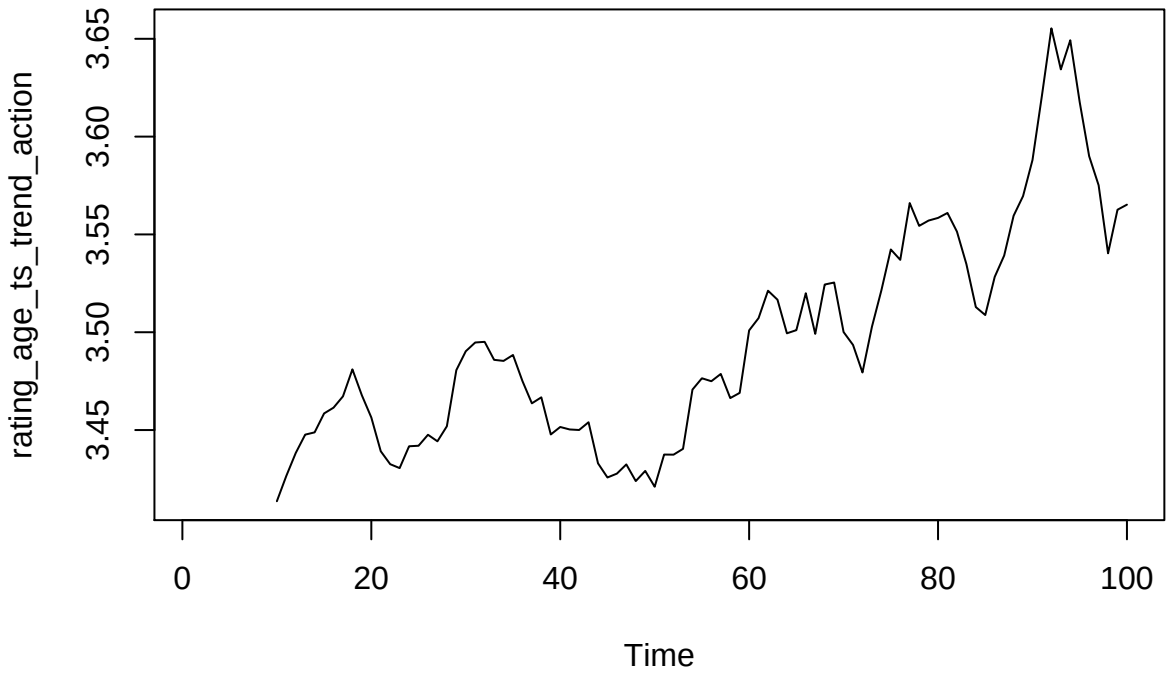
```
summary(ur.kpss(rating_age_ts_action))
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.9707
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

After testing stationarity, we can choose a suitable model.

Extract trend

```
rating_age_ts_trend_action<-SMA(rating_age_ts_action)
plot.ts(rating_age_ts_trend_action)
```



The trend for action movies is obviously upward.

Test trend

```
trend.test(rating_age_ts_action, R=1)
```

```
## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  rating_age_ts_action and time(rating_age_ts_action)
## S = 101934, p-value = 6.54e-05
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.3883372
```

Test trend for all genres

```
rating_age_ts<-spread(ratings_genres_age_ts,genres,mean_rating)[,-1]
trend.p.value<-function(x){
  return(as.numeric(unlist(trend.test(x, R=1))[2]))
}
trend.rho<-function(x){
  return(as.numeric(unlist(trend.test(x, R=1))[3]))
}
genres_trend.p.value<-lapply(rating_age_ts,trend.p.value)
```

```
## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties
```

```
genres_trend.rho<-lapply(rating_age_ts,trend.rho)
```

```
## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties
```

```
## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties

## Warning in cor.test.default(x, time(x), alternative = "two.sided", method =
## "spearman"): Cannot compute exact p-value with ties
```

Trend is not significant

```
names(genres_trend.p.value[genres_trend.p.value>0.05])
```

```
##  [1] "Adventure"   "Children"    "Comedy"      "Documentary" "Drama"
##  [6] "Fantasy"     "Film-Noir"   "IMAX"        "Musical"     "Mystery"
## [11] "Romance"     "War"         "Western"
```

Trend is not significant

Trend is upward

```
names(genres_trend.p.value[(genres_trend.p.value<=0.05)&(genres_trend.rho>0)])
```

```
## [1] "Action"   "Crime"    "Horror"   "Sci-Fi"   "Thriller"
```

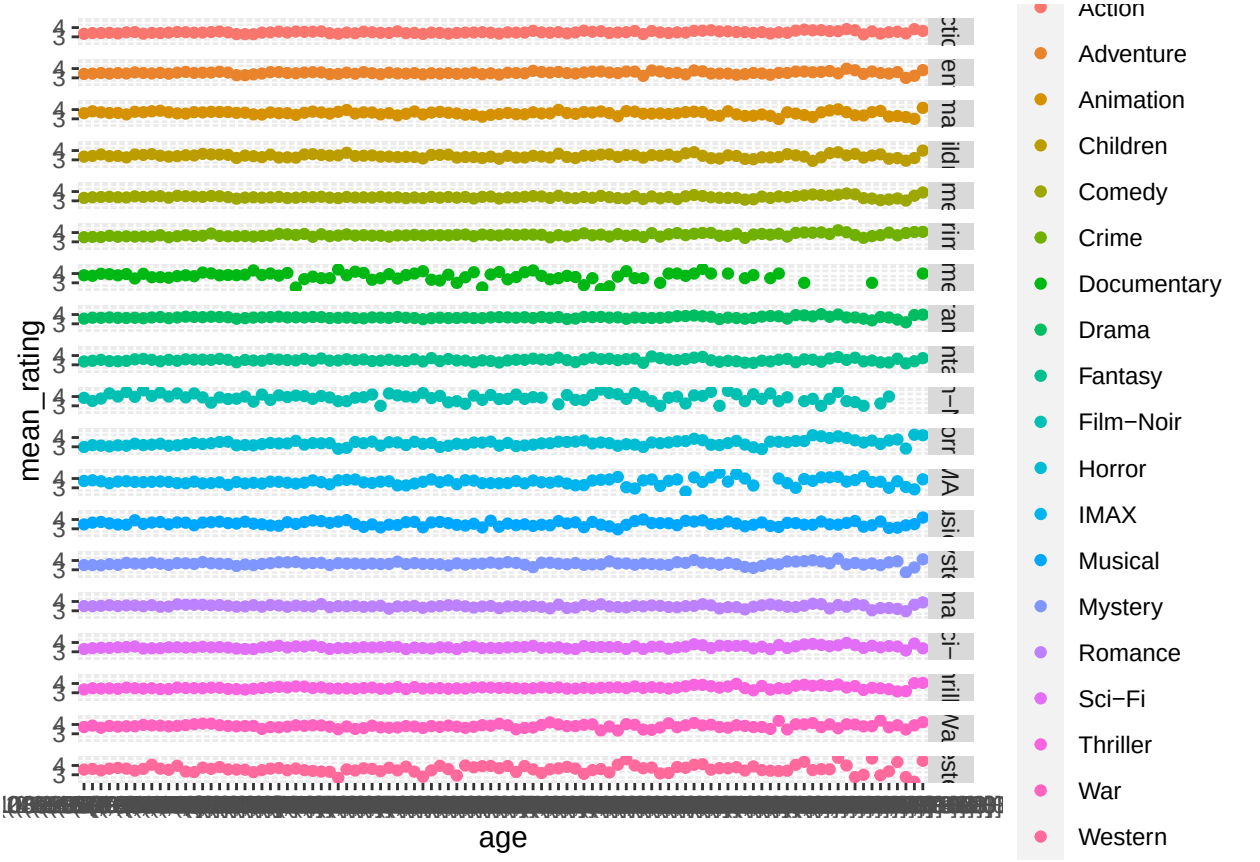
Trend is downward

```
names(genres_trend.p.value[(genres_trend.p.value<=0.05)&(genres_trend.rho<=0)])
```

```
## [1] "Animation"
```

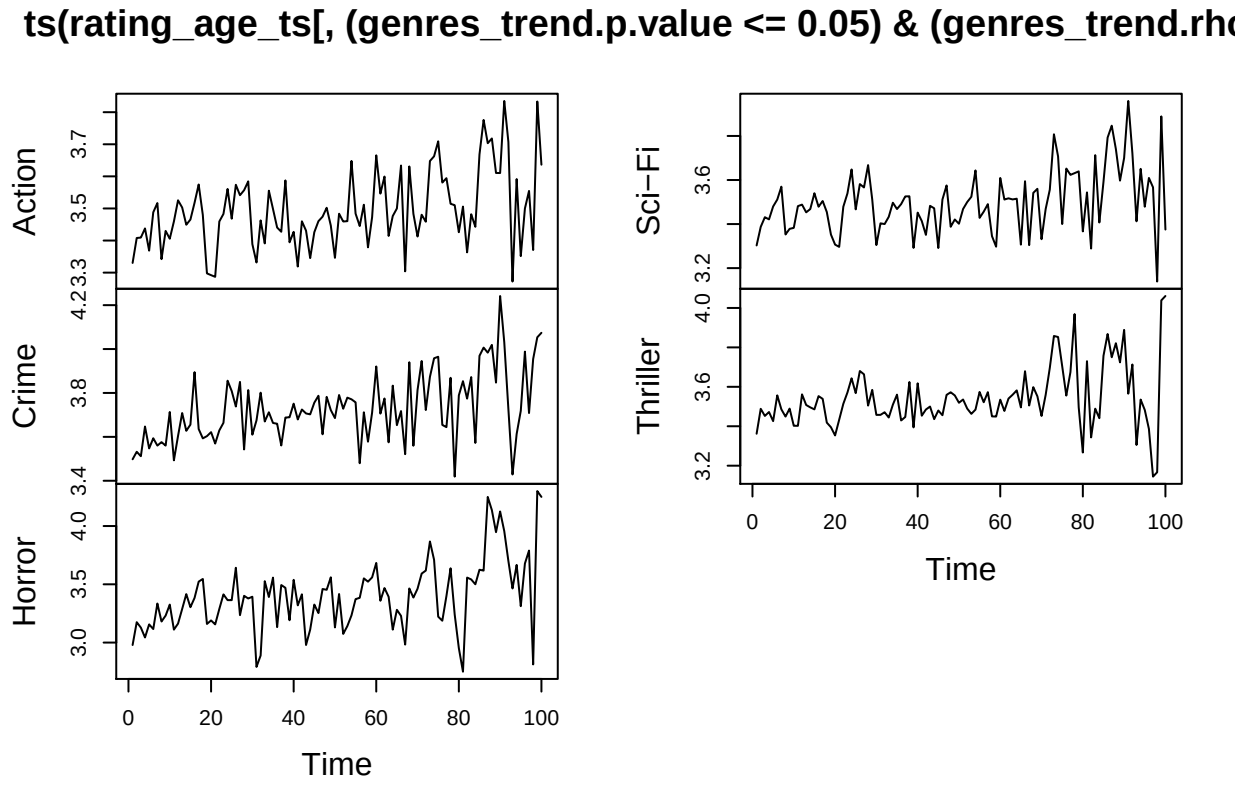
Plot trends for all genres

```
ggplot(ratings_genres_age_ts, aes(x=age, y=mean_rating, colour=genres))+
  geom_point() +
  facet_grid(genres ~.)
```



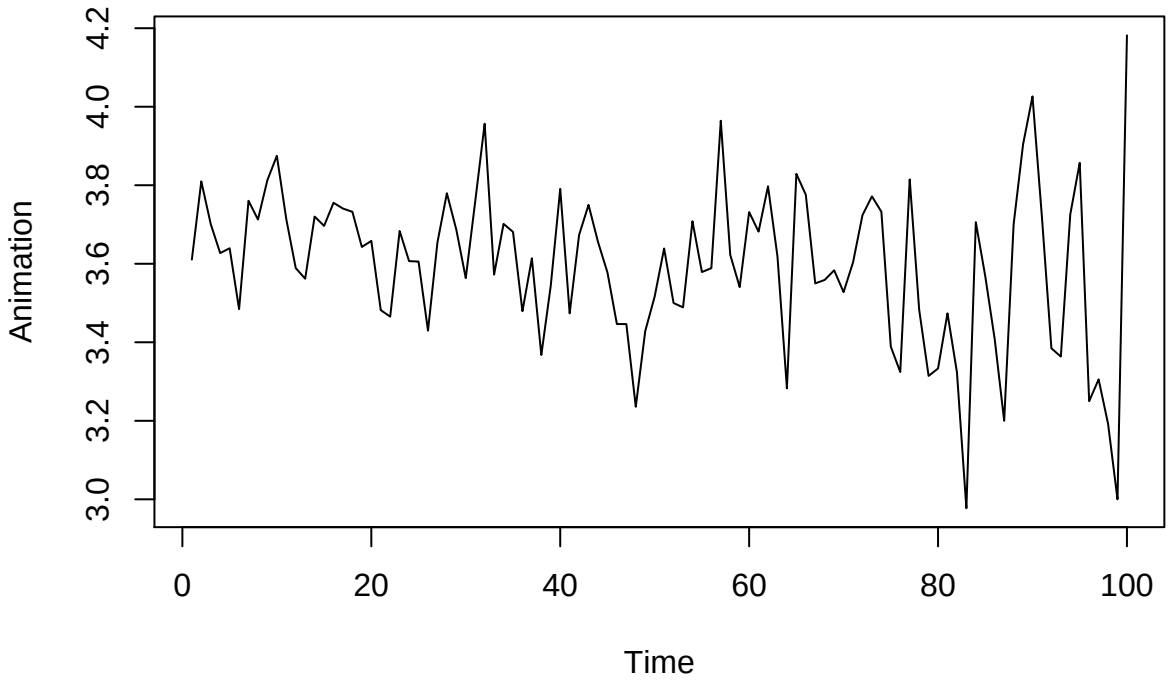
Plot upward and significant trend

```
plot(ts(rating_age_ts[,(genres_trend.p.value<=0.05)&(genres_trend.rho>0)]))
```



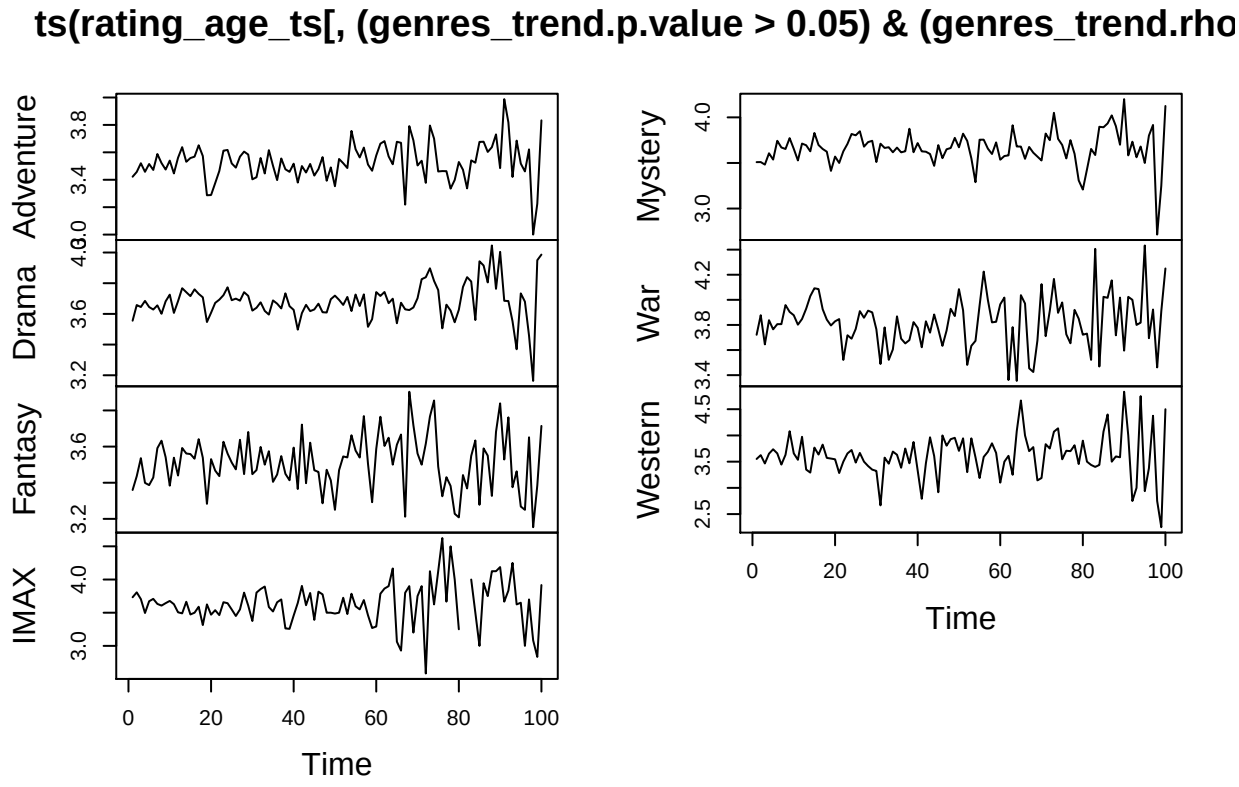
Plot downward and significant trend

```
plot(ts(rating_age_ts[,(genres_trend.p.value<=0.05)&(genres_trend.rho<=0)]))
```



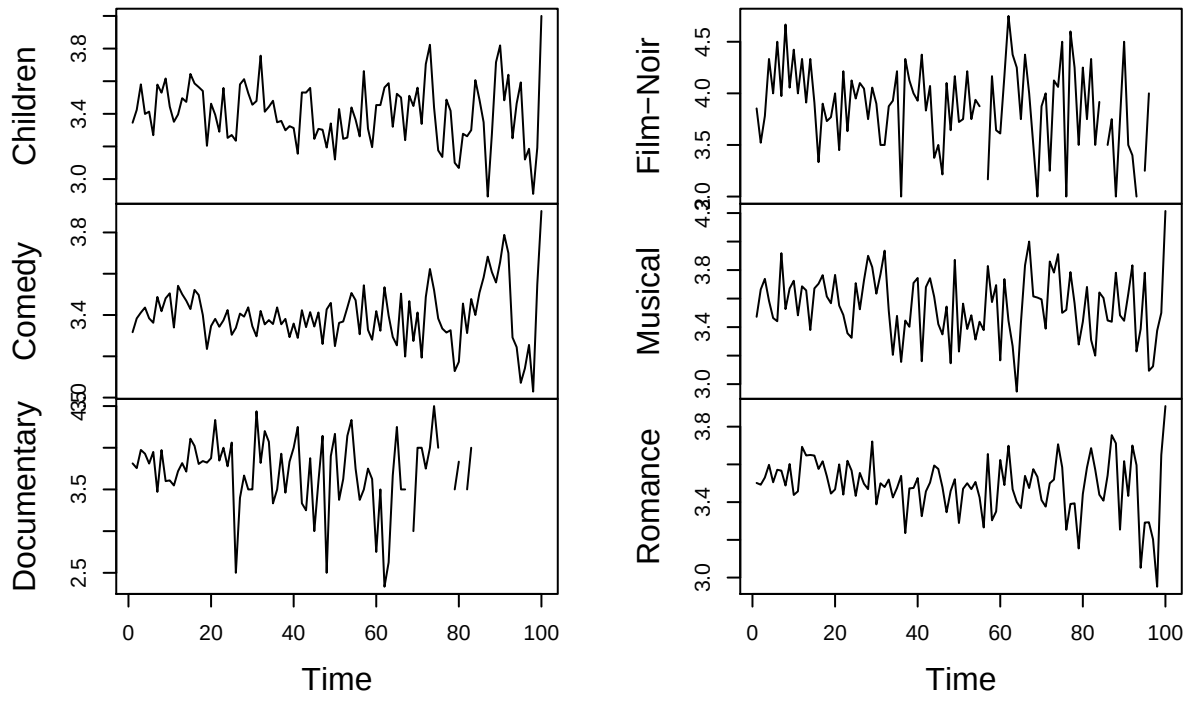
Plot upward and insignificant trend (Compared to the upward and significant one)

```
plot(ts(rating_age_ts[,(genres_trend.p.value>0.05)&(genres_trend.rho>0)]))
```



Plot downward and insignificant trend (Compared to the downward and significant one)

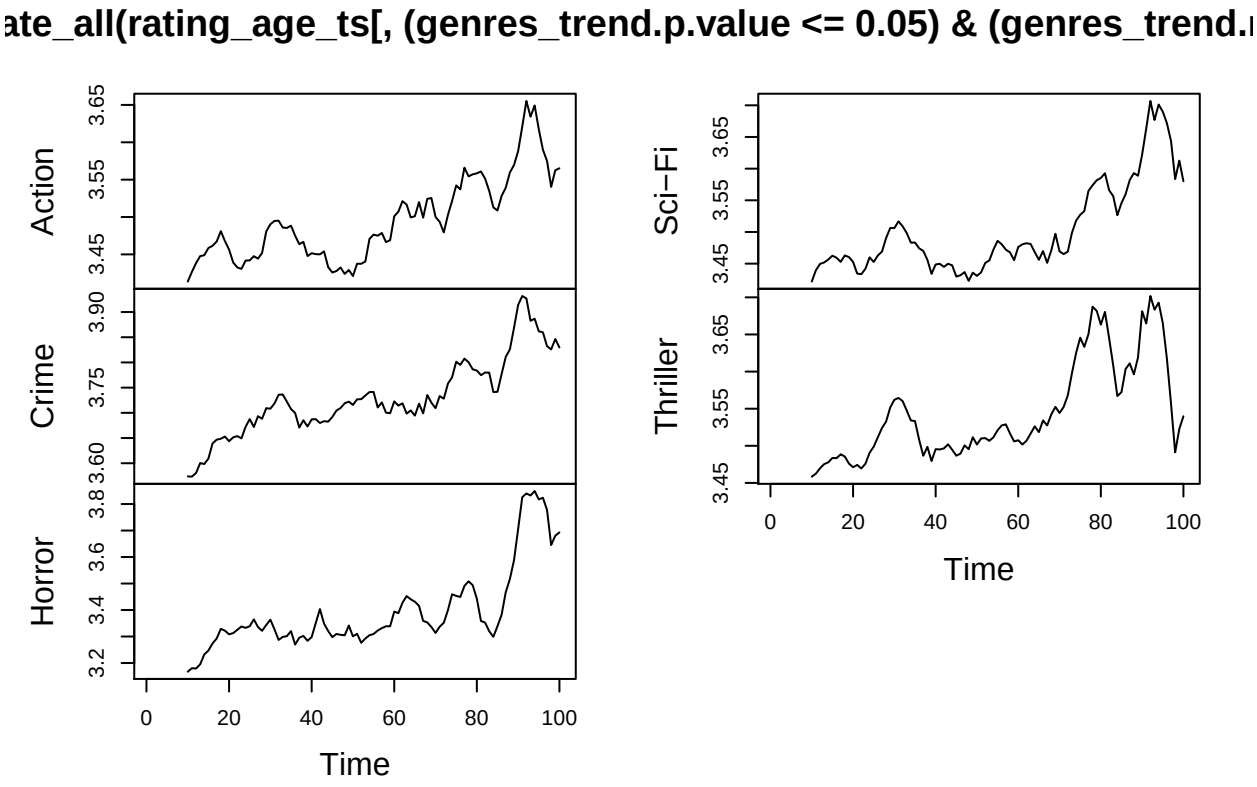
```
plot(ts(rating_age_ts[,(genres_trend.p.value>0.05)&(genres_trend.rho<=0)]))
```



Plot extracted upward and significant trend

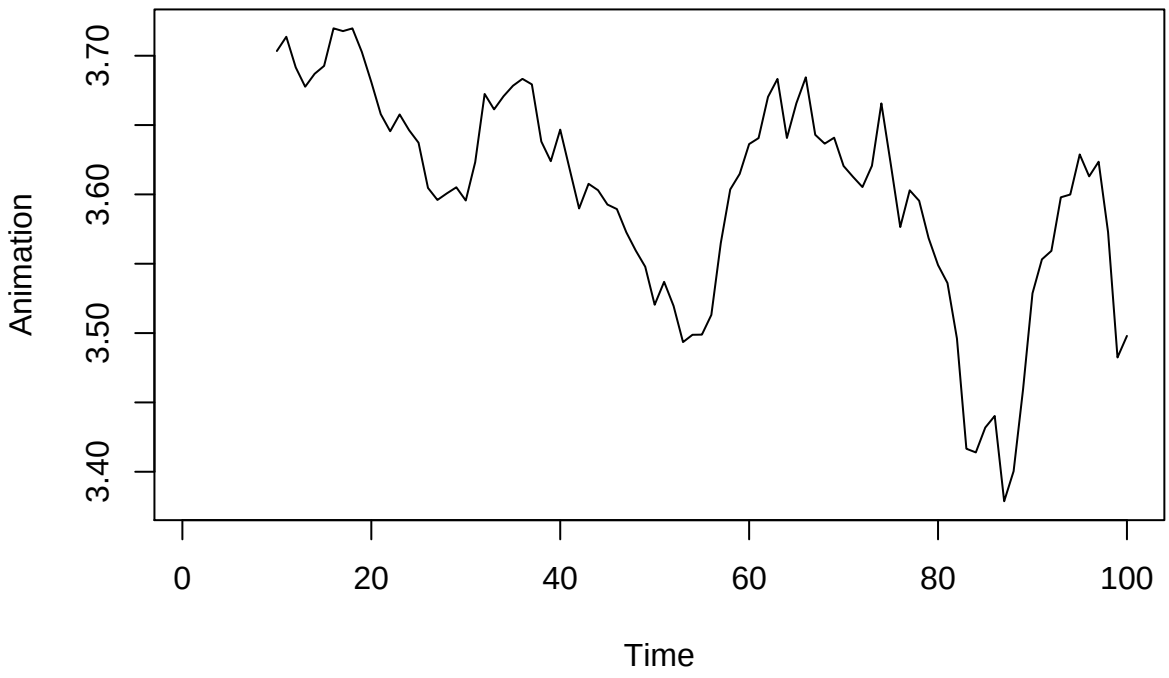
```
#impute missing data (use mean for each genre)
rating_age_ts<-mutate_all(rating_age_ts,impute,mean)
```

```
plot(ts(mutate_all(rating_age_ts[,(genres_trend.p.value<=0.05)&(genres_trend.rho>0)],SMA)))
```



Plot extracted downward and significant trend

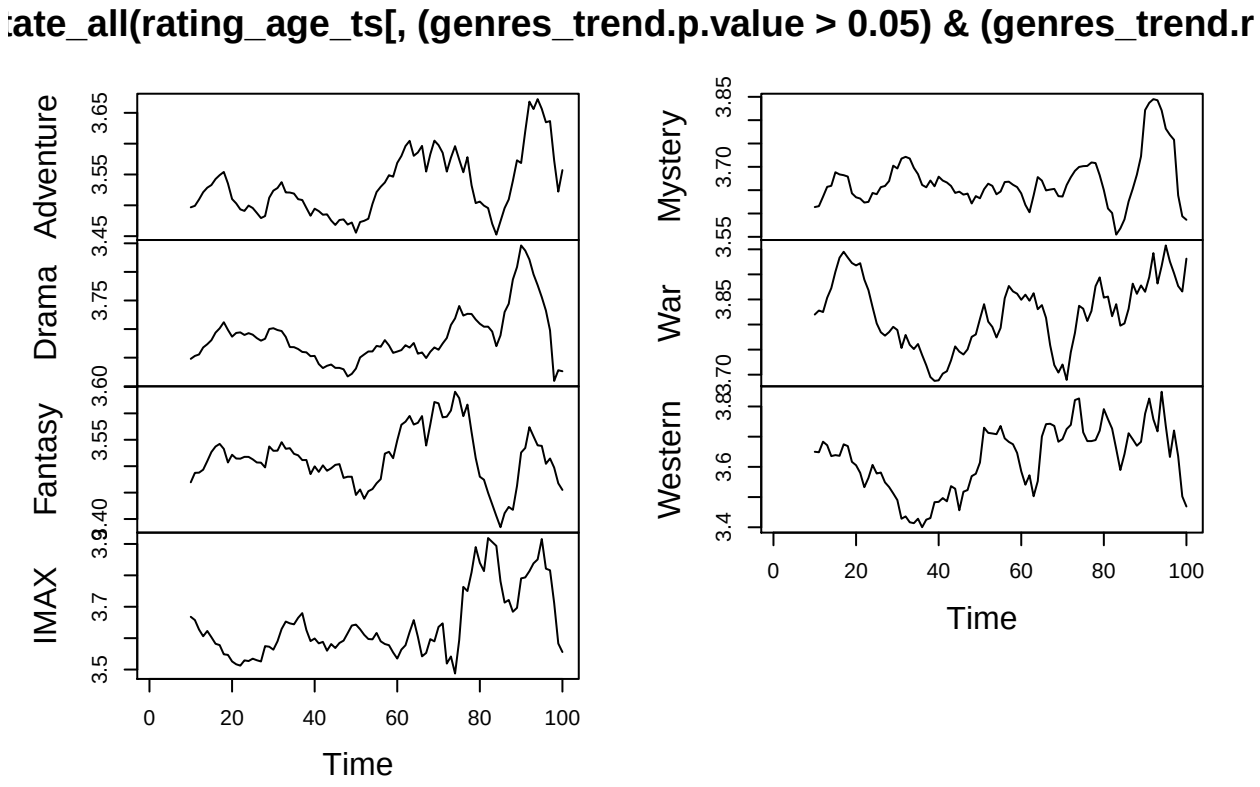
```
plot(ts(mutate_all(rating_age_ts[,(genres_trend.p.value<=0.05)&(genres_trend.rho<=0)],SMA)))
```



For the siginificant trend, both upward and downward are obvious.

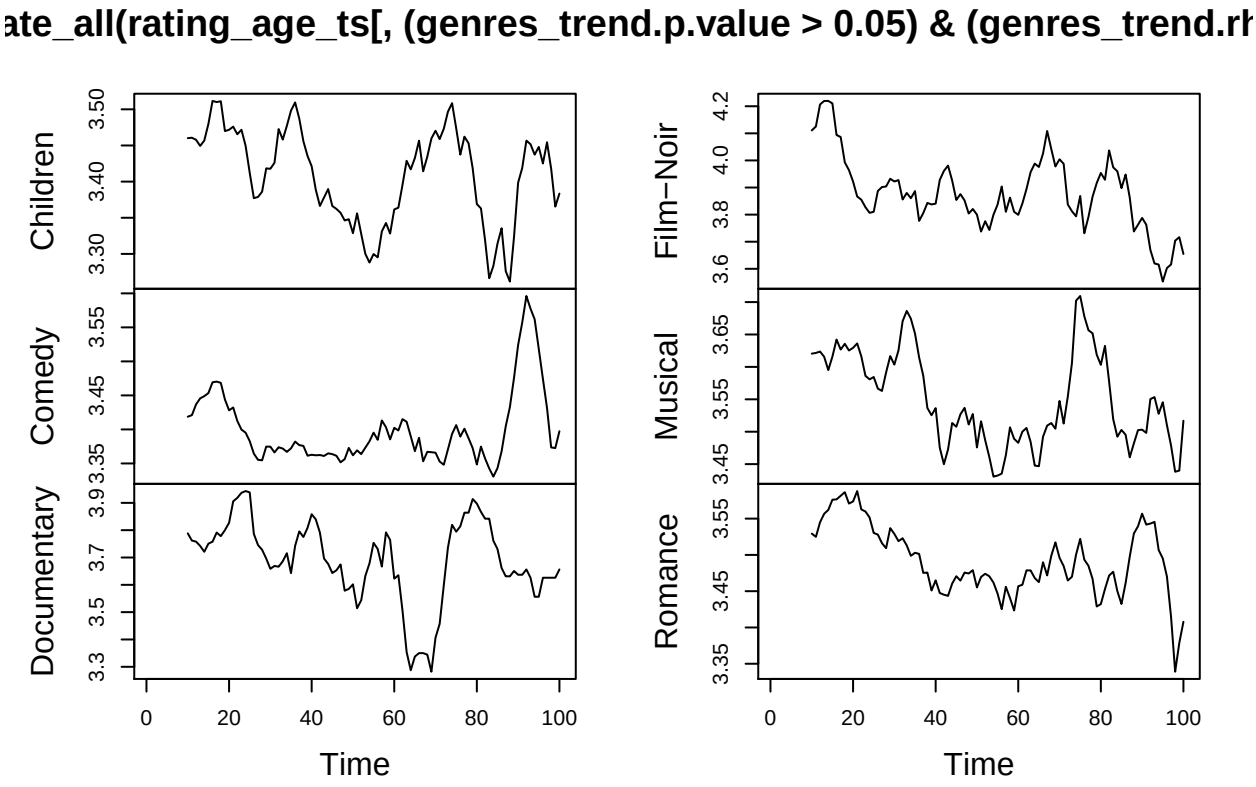
Plot extracted upward and not significant trend

```
plot(ts(mutate_all(rating_age_ts[,(genres_trend.p.value>0.05)&(genres_trend.rho>0)],SMA)))
```



Plot extracted downward and not significant trend

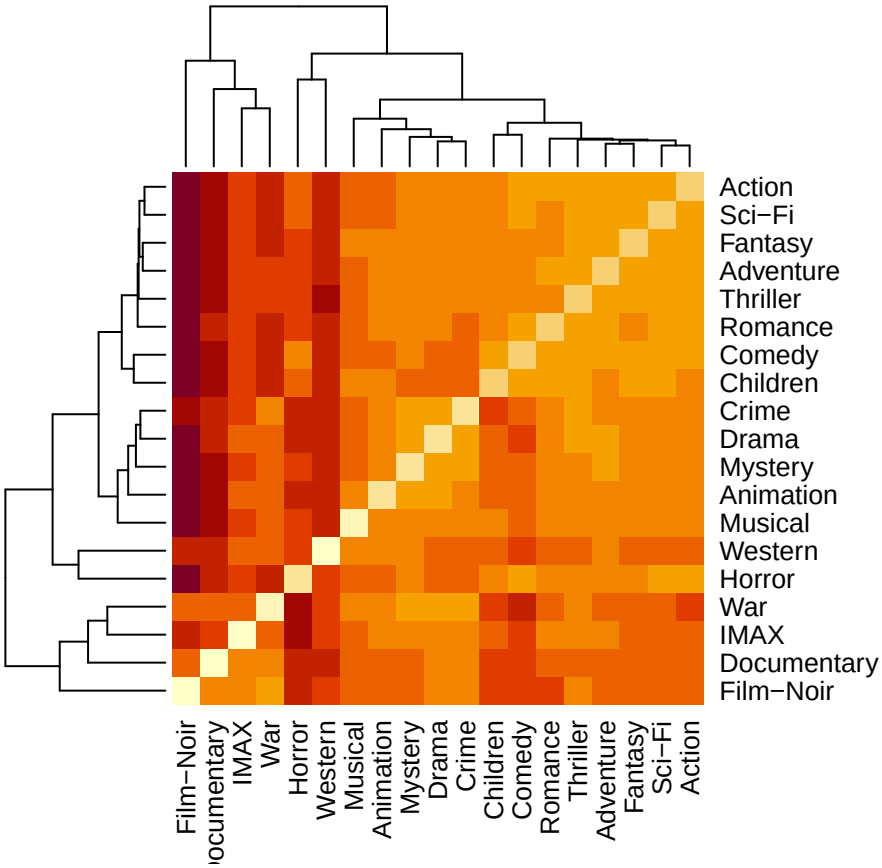
```
plot(ts(mutate_all(rating_age_ts[,(genres_trend.p.value>0.05)&(genres_trend.rho<=0)],SMA)))
```



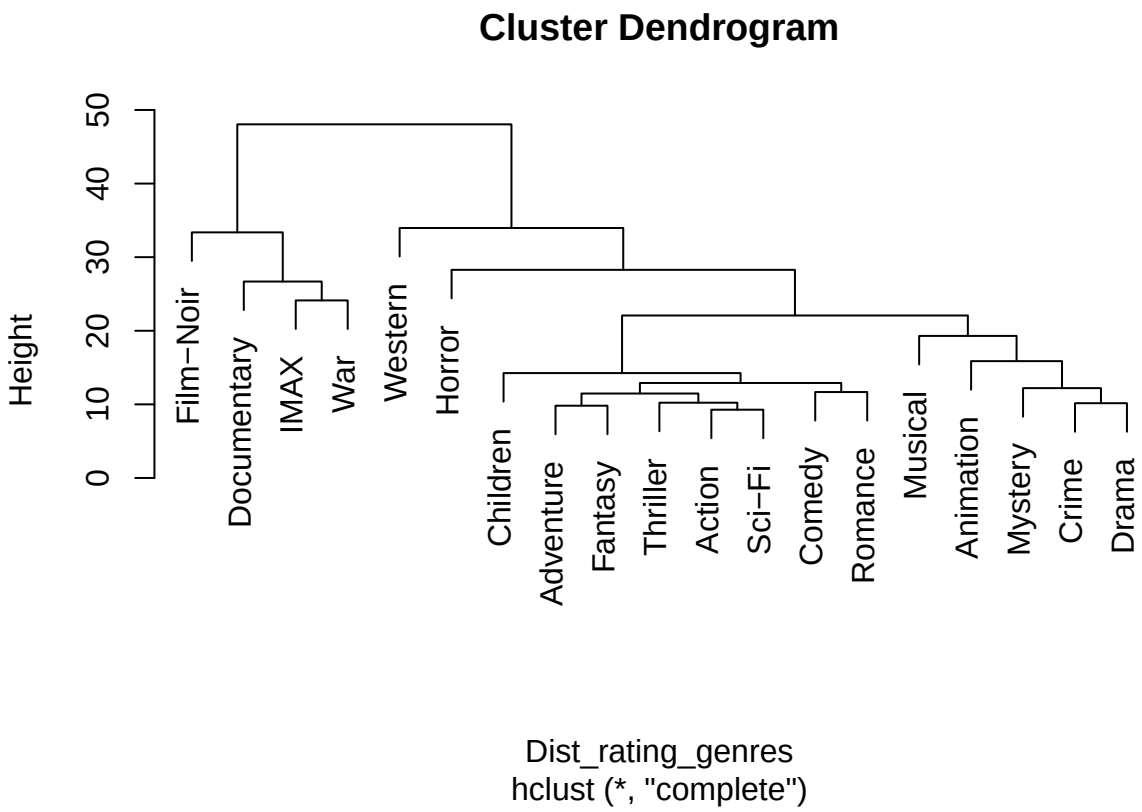
We cannot extract obvious trend because they are not significant.

Clustering

```
Dist_rating_genres<-diss(rating_age_ts,"DTWARP")
heatmap(as.matrix(Dist_rating_genres))
```




```
Clust_genres<-hclust(Dist_rating_genres)
plot(Clust_genres)
```


Cluster Dendrogram
Height
0
10
20
30
40
50
Film−Noir
Documentary
IMAX
War
Western
Horror
Children
Adventure
Fantasy
Thriller
Action
Sci−Fi
Comedy
Romance
Musical
Animation
Mystery
Crime
Drama
Dist_rating_genres
hclust (*, "complete")