airline model

linyiguo

2019/7/25

Data Exploratory

```r
library(forecast)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

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

```
## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```r
library(tseries)
library(ggfortify)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggfortify':
##   method                 from
##   autoplot.Arima         forecast
##   autoplot.acf           forecast
##   autoplot.ar            forecast
##   autoplot.bats          forecast
##   autoplot.decomposed.ts forecast
##   autoplot.ets           forecast
##   autoplot.forecast      forecast
##   autoplot.stl           forecast
##   autoplot.ts            forecast
##   fitted.ar              forecast
##   fortify.ts             forecast
##   residuals.ar           forecast
```

```r
data("AirPassengers")
data_ap <- AirPassengers

data_ap
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

```r
sum(is.na(data_ap))
```

```
## [1] 0
```

```r
frequency(data_ap)
```

```
## [1] 12
```

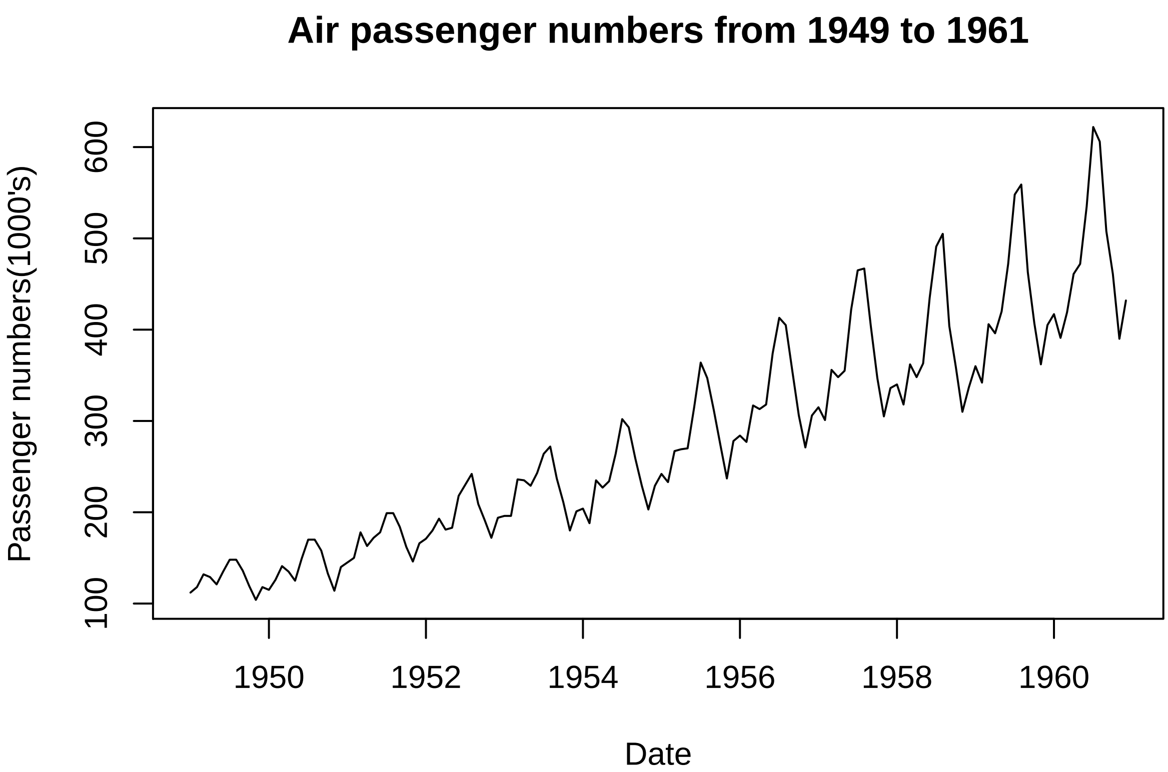
```r
cycle(data_ap)
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949   1   2   3   4   5   6   7   8   9  10  11  12
## 1950   1   2   3   4   5   6   7   8   9  10  11  12
## 1951   1   2   3   4   5   6   7   8   9  10  11  12
## 1952   1   2   3   4   5   6   7   8   9  10  11  12
## 1953   1   2   3   4   5   6   7   8   9  10  11  12
## 1954   1   2   3   4   5   6   7   8   9  10  11  12
## 1955   1   2   3   4   5   6   7   8   9  10  11  12
## 1956   1   2   3   4   5   6   7   8   9  10  11  12
## 1957   1   2   3   4   5   6   7   8   9  10  11  12
## 1958   1   2   3   4   5   6   7   8   9  10  11  12
## 1959   1   2   3   4   5   6   7   8   9  10  11  12
## 1960   1   2   3   4   5   6   7   8   9  10  11  12
```

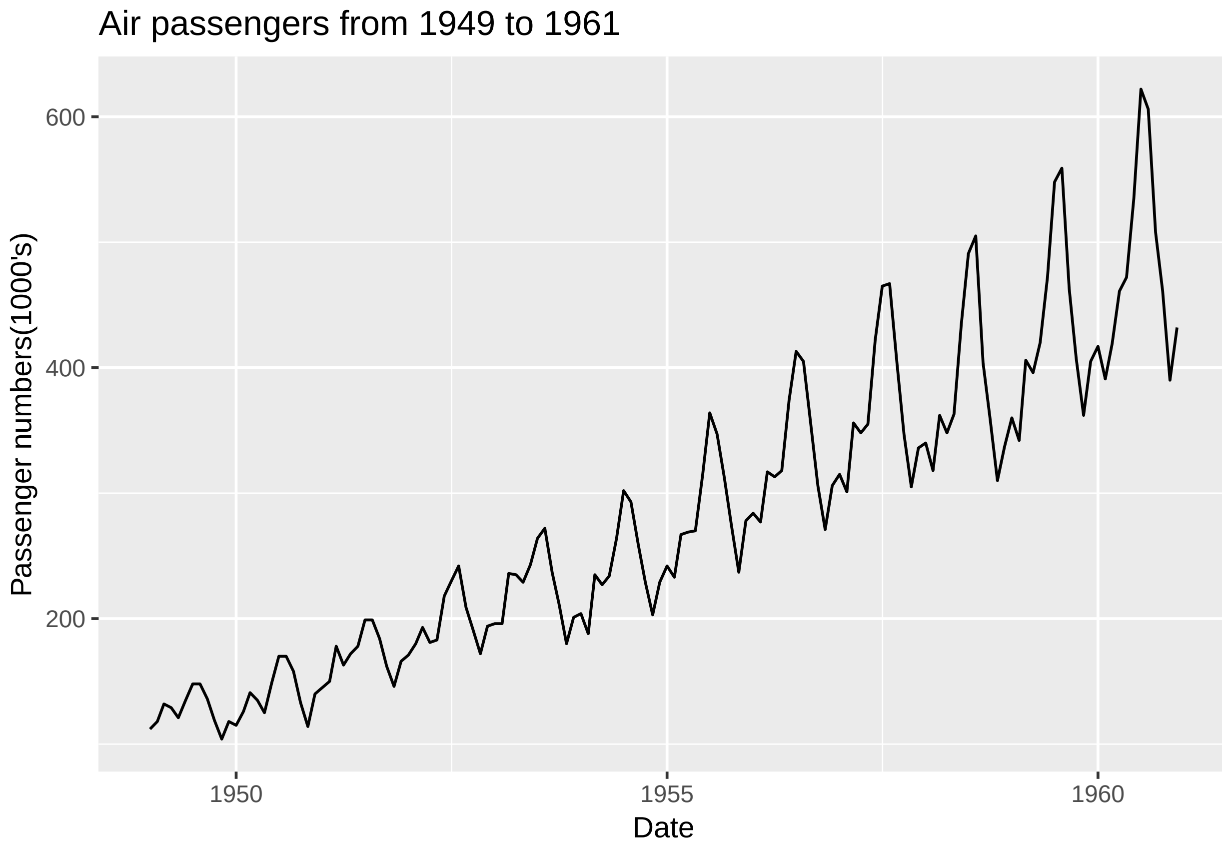
```r
summary(data_ap)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   104.0   180.0   265.5   280.3   360.5   622.0
```

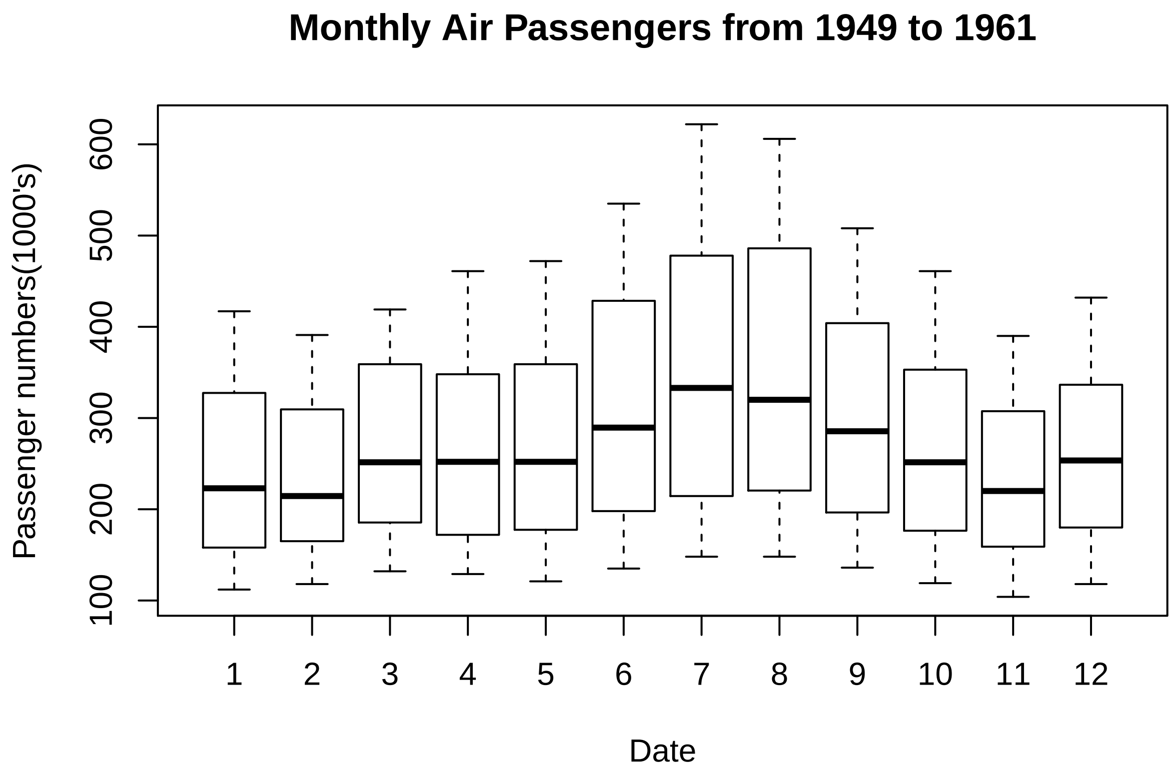
```r
plot(data_ap, xlab="Date", ylab="Passenger numbers(1000's)",
     main = "Air passenger numbers from 1949 to 1961")
```



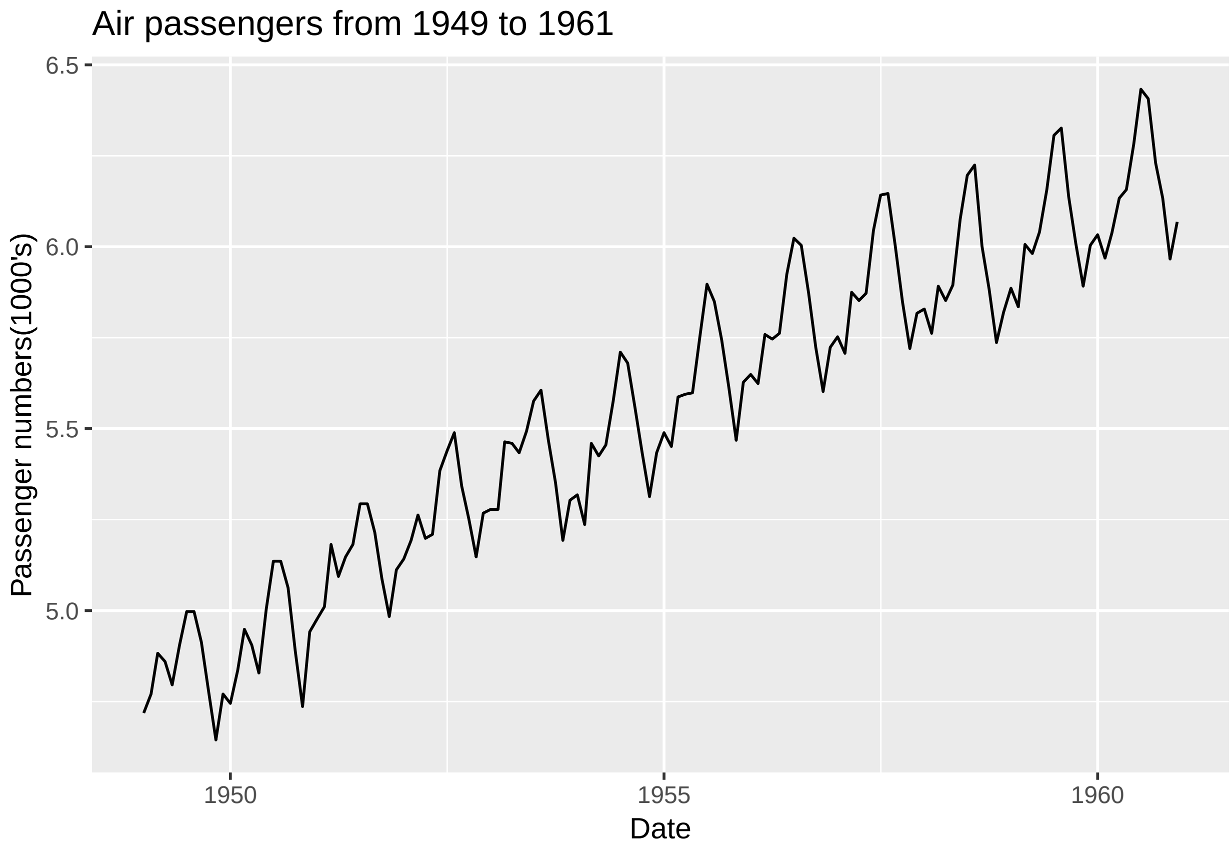

```r
autoplot(data_ap)+labs(x="Date", y="Passenger numbers(1000's)",
                       title = "Air passengers from 1949 to 1961")
```




```r
boxplot(data_ap~cycle(data_ap), xlab="Date", ylab="Passenger numbers(1000's)",
        main="Monthly Air Passengers from 1949 to 1961")
```




```
autoplot(log(data_ap))+labs(x="Date", y="Passenger numbers(1000's)",
                            title = "Air passengers from 1949 to 1961")
```



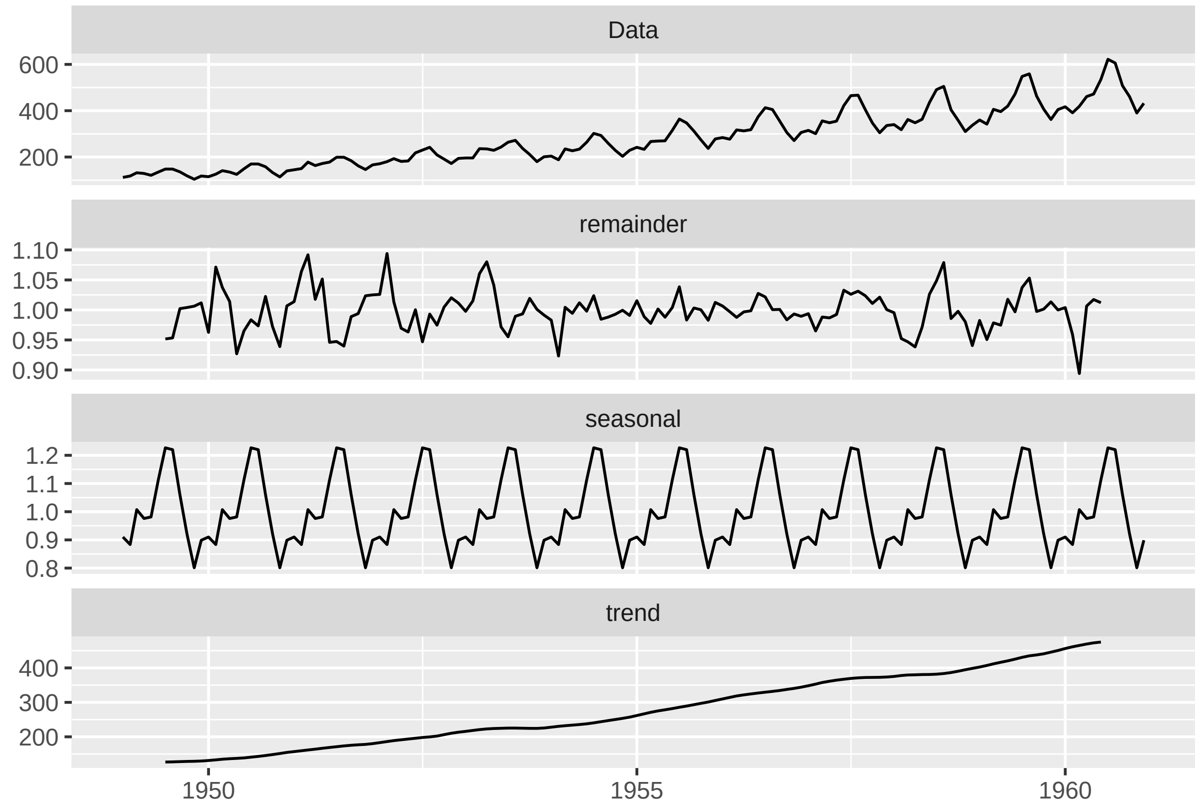
Note The amplitude of the original data is becoming larger as time goes, so we need to take `log` if want to use `additive` in decomposation.

Time Series Decomposition

```
decompose_ap <- decompose(data_ap, "multiplicative")
#decompose_ap'class is decompose.ts
autoplot(decompose_ap)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

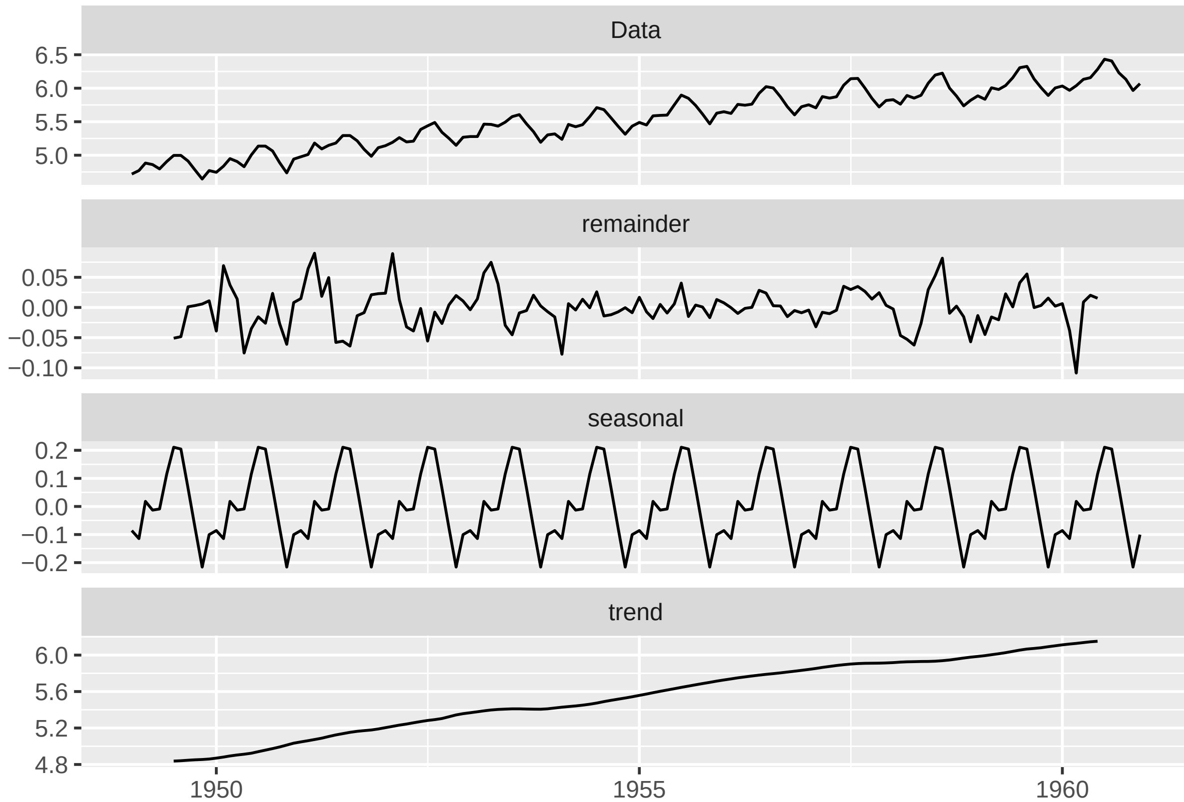
```
## Warning: Removed 24 rows containing missing values (geom_path).
```




```
decompose_ap1 <- decompose(log(data_ap), "additive")
autoplot(decompose_ap1)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped

## Warning: Removed 24 rows containing missing values (geom_path).
```



Test Stationarity of The Time Series

1. ADF(Augmented Dickey-Fuller) Test

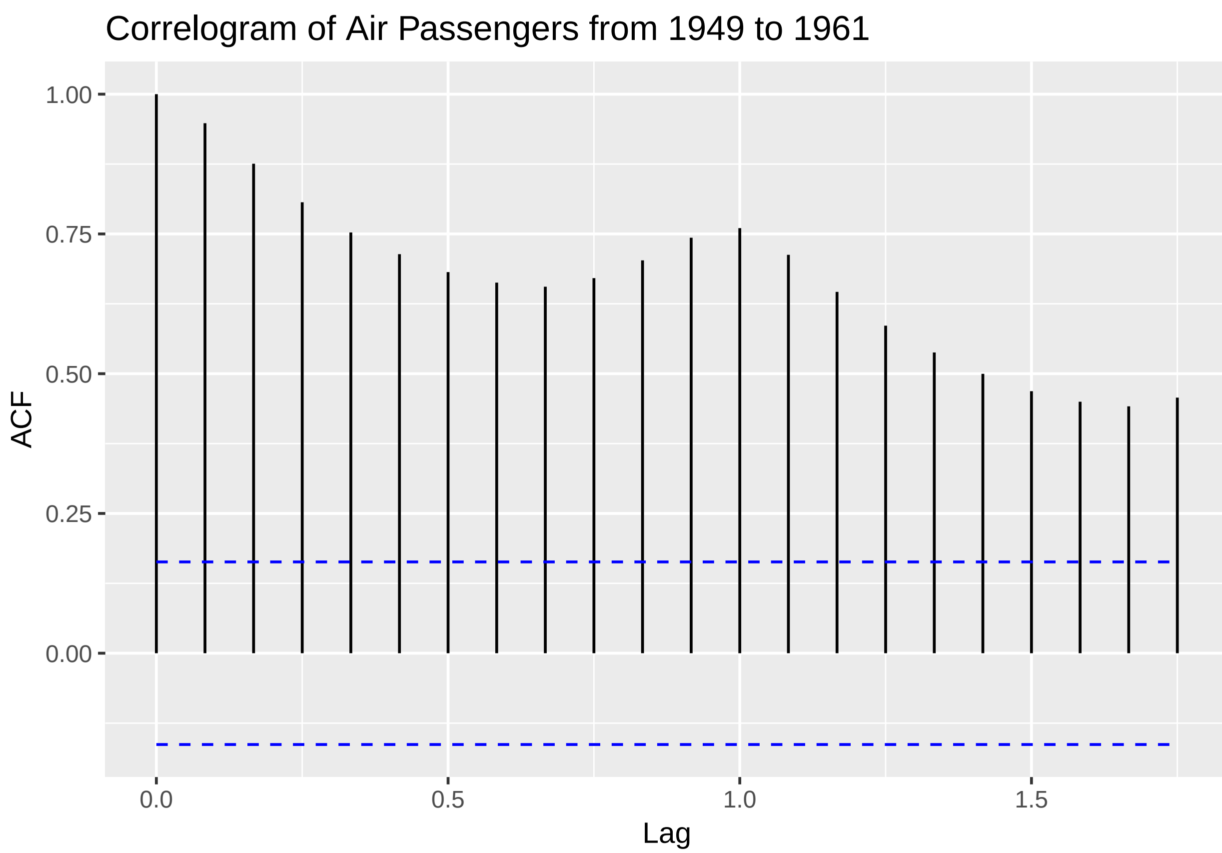
```
adf.test(data_ap)
```

```
## Warning in adf.test(data_ap): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_ap
## Dickey-Fuller = -7.3186, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

2. ACF(Autocorrelation) Test

```
autoplot(acf(data_ap, plot=F))+labs(title="Correlogram of Air Passengers from 1949 to 1961")
```



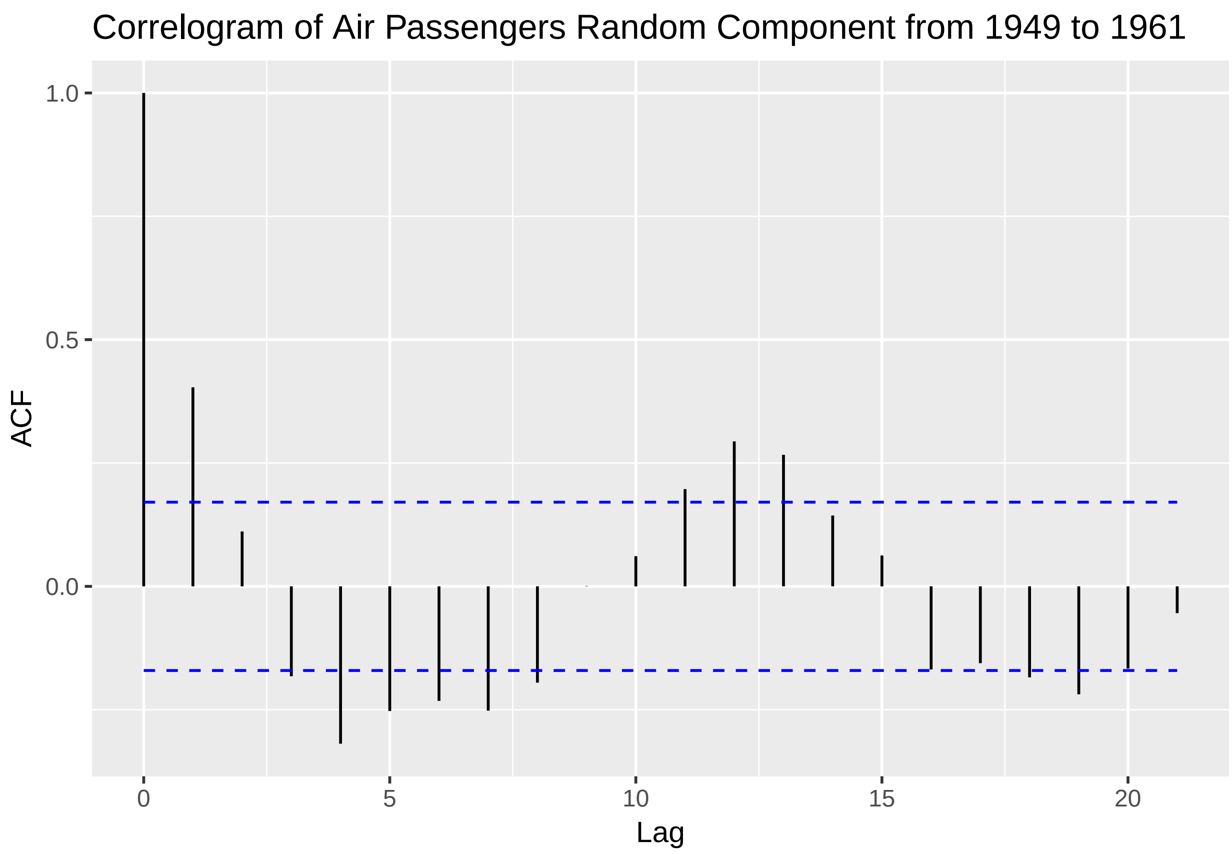
If the autocorrelation crosses the dashed blue line, it means that specific lag is significantly correlated with current series.

```
decompose_ap$random # the first and last 6 residuals are missing
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 1949        NA        NA        NA        NA        NA        NA 0.9516643
## 1950 0.9626030 1.0714668 1.0374474 1.0140476 0.9269030 0.9650406 0.9835566
## 1951 1.0138446 1.0640180 1.0918541 1.0176651 1.0515825 0.9460444 0.9474041
## 1952 1.0258814 1.0939696 1.0134734 0.9695596 0.9632673 1.0003735 0.9468562
## 1953 0.9976684 1.0151646 1.0604644 1.0802327 1.0413329 0.9718056 0.9551933
## 1954 0.9829785 0.9232032 1.0044417 0.9943899 1.0119479 0.9978740 1.0237753
## 1955 1.0154046 0.9888241 0.9775844 1.0015732 0.9878755 1.0039635 1.0385512
## 1956 1.0066157 0.9970250 0.9876248 0.9968224 0.9985644 1.0275560 1.0217685
## 1957 0.9937293 0.9649918 0.9881769 0.9867637 0.9924177 1.0328601 1.0261250
## 1958 0.9954212 0.9522762 0.9469115 0.9383993 0.9715785 1.0261340 1.0483841
## 1959 0.9825176 0.9505736 0.9785278 0.9746440 1.0177637 0.9968613 1.0373136
## 1960 1.0039279 0.9590794 0.8940857 1.0064948 1.0173588 1.0120790        NA
##            Aug       Sep       Oct       Nov       Dec
## 1949 0.9534014 1.0022198 1.0040278 1.0062701 1.0118119
## 1950 0.9733720 1.0225047 0.9721928 0.9389527 1.0067914
## 1951 0.9397599 0.9888637 0.9938809 1.0235337 1.0250824
## 1952 0.9931171 0.9746302 1.0046687 1.0202797 1.0115407
## 1953 0.9894989 0.9934337 1.0192680 1.0009392 0.9915039
## 1954 0.9845184 0.9881036 0.9927613 0.9995143 0.9908692
## 1955 0.9831117 1.0032501 1.0003084 0.9827720 1.0125535
```

```
## 1956 1.0004765 1.0008730 0.9835071 0.9932761 0.9894251
## 1957 1.0312668 1.0236147 1.0108432 1.0212995 1.0005263
## 1958 1.0789695 0.9856540 0.9977971 0.9802940 0.9405687
## 1959 1.0531001 0.9974447 1.0013371 1.0134608 0.9999192
## 1960        NA        NA        NA        NA        NA
```

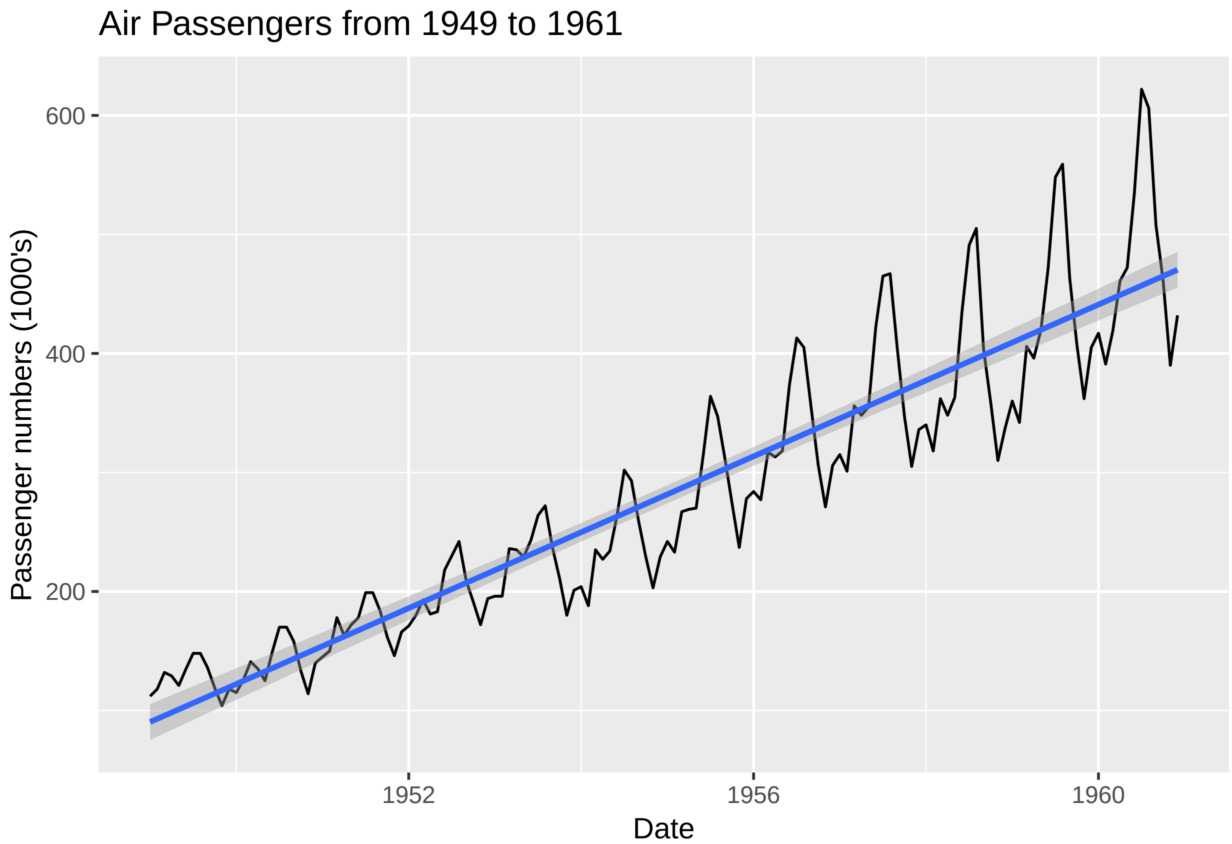
```
autoplot(acf(decompose_ap$random[7:138],plot=FALSE))+ labs(title="Correlogram of Air Passengers Random C
```



Fit a Time Series Model

1. Linear Model

```
data_ap_dm <- cbind(as.numeric(time(data_ap)),as.numeric(data_ap))
colnames(data_ap_dm) <- c('t','passengers')
ggplot(data_ap_dm, aes(x=t,y=passengers)) + geom_line() + geom_smooth(method = "lm") + labs(x ="Date", y
```



2. Arima Model

`ggtsdiag` is from package `ggfortify` and used to perform diagnostics of residuals.

```
arima_ap <- auto.arima(data_ap)
arima_ap
```

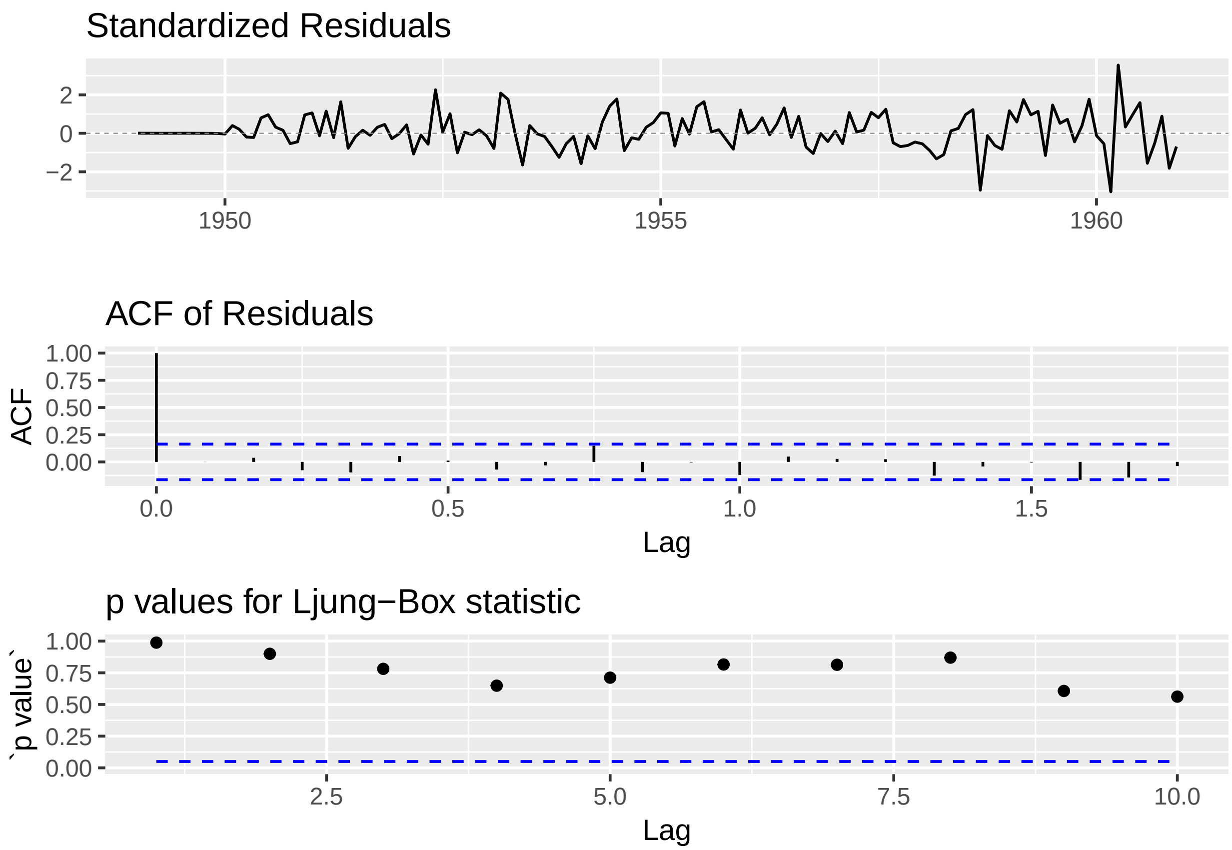
```
## Series: data_ap
## ARIMA(2,1,1)(0,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1
##       0.5960  0.2143  -0.9819
## s.e.  0.0888  0.0880   0.0292
##
## sigma^2 estimated as 132.3:  log likelihood=-504.92
## AIC=1017.85   AICc=1018.17   BIC=1029.35
```

```
arima_ap1 <- auto.arima(log(data_ap))
arima_ap1
```

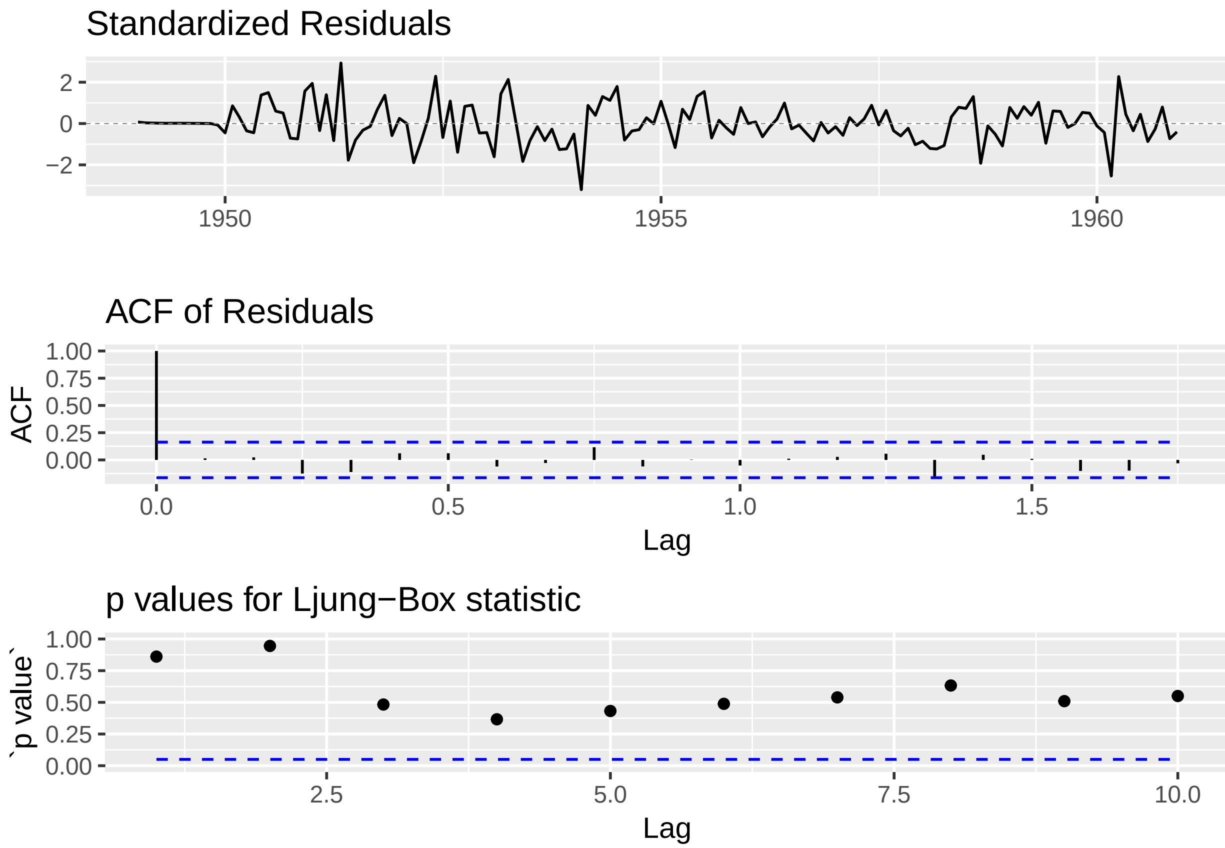
```
## Series: log(data_ap)
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
```

```
##           ma1     sma1
##       -0.4018  -0.5569
## s.e.   0.0896   0.0731
##
## sigma^2 estimated as 0.001371:  log likelihood=244.7
## AIC=-483.4   AICc=-483.21   BIC=-474.77
```

```r
ggtsdiag(arima_ap)
```

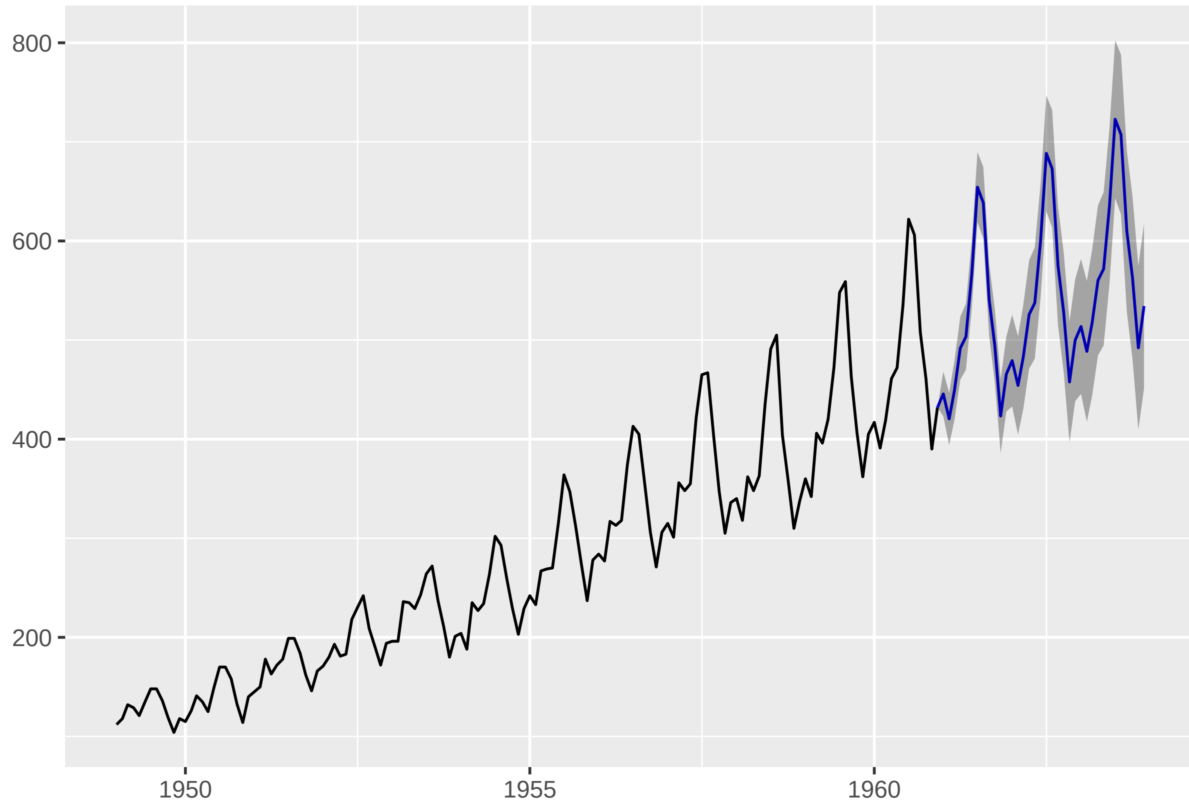



```r
ggtsdiag(arima_ap1)
```

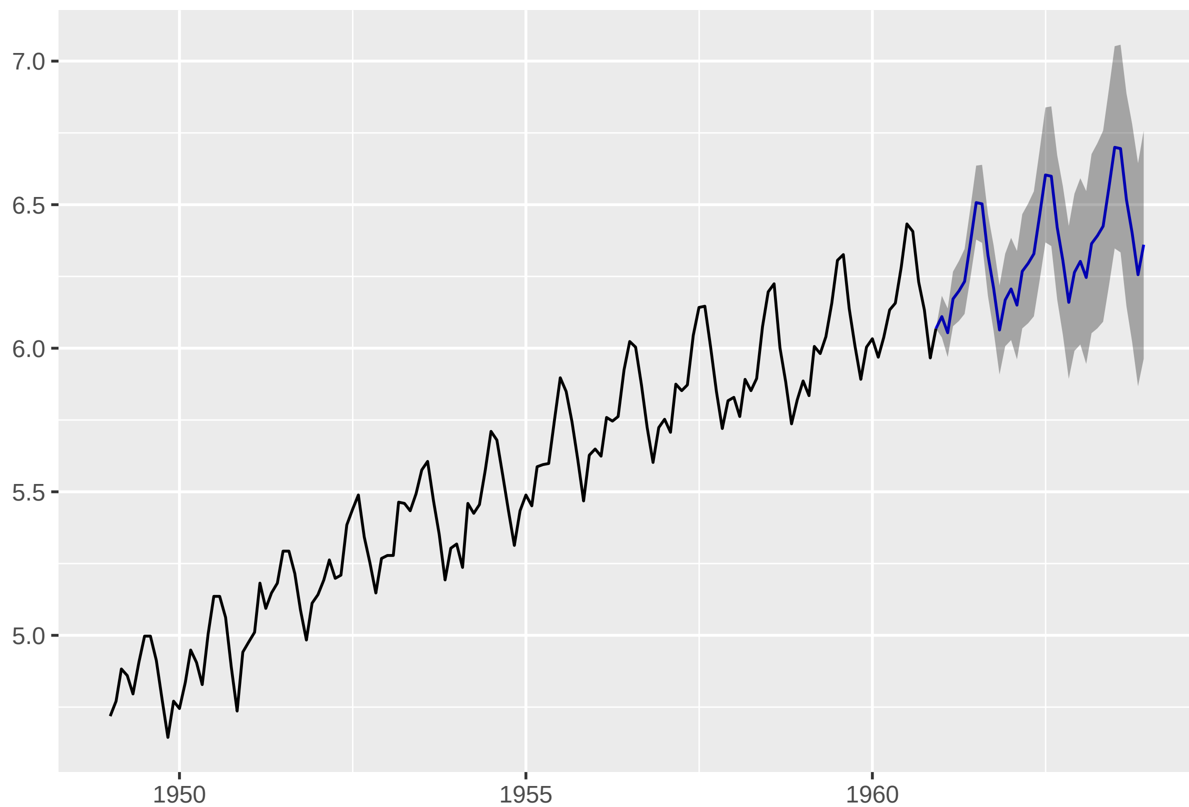


Calculate Forecast

```
forecast_ap <- forecast(arima_ap, level=c(95), h=36)
autoplot(forecast_ap)
```




```
forecast_ap1 <- forecast(arima_ap1, level=c(95), h=36)
autoplot(forecast_ap1)
```




```r
as.numeric(forecast_ap1$x)
```

```
##   [1] 4.718499 4.770685 4.882802 4.859812 4.795791 4.905275 4.997212
##   [8] 4.997212 4.912655 4.779123 4.644391 4.770685 4.744932 4.836282
##  [15] 4.948760 4.905275 4.828314 5.003946 5.135798 5.135798 5.062595
##  [22] 4.890349 4.736198 4.941642 4.976734 5.010635 5.181784 5.093750
##  [29] 5.147494 5.181784 5.293305 5.293305 5.214936 5.087596 4.983607
##  [36] 5.111988 5.141664 5.192957 5.262690 5.198497 5.209486 5.384495
##  [43] 5.438079 5.488938 5.342334 5.252273 5.147494 5.267858 5.278115
##  [50] 5.278115 5.463832 5.459586 5.433722 5.493061 5.575949 5.605802
##  [57] 5.468060 5.351858 5.192957 5.303305 5.318120 5.236442 5.459586
##  [64] 5.424950 5.455321 5.575949 5.710427 5.680173 5.556828 5.433722
##  [71] 5.313206 5.433722 5.488938 5.451038 5.587249 5.594711 5.598422
##  [78] 5.752573 5.897154 5.849325 5.743003 5.613128 5.468060 5.627621
##  [85] 5.648974 5.624018 5.758902 5.746203 5.762051 5.924256 6.023448
##  [92] 6.003887 5.872118 5.723585 5.602119 5.723585 5.752573 5.707110
##  [99] 5.874931 5.852202 5.872118 6.045005 6.142037 6.146329 6.001415
## [106] 5.849325 5.720312 5.817111 5.828946 5.762051 5.891644 5.852202
## [113] 5.894403 6.075346 6.196444 6.224558 6.001415 5.883322 5.736572
## [120] 5.820083 5.886104 5.834811 6.006353 5.981414 6.040255 6.156979
## [127] 6.306275 6.326149 6.137727 6.008813 5.891644 6.003887 6.033086
## [134] 5.968708 6.037871 6.133398 6.156979 6.282267 6.432940 6.406880
## [141] 6.230481 6.133398 5.966147 6.068426
```

```r
exp(as.numeric(forecast_ap1$x)) # x is original series
```

```
##   [1] 112 118 132 129 121 135 148 148 136 119 104 118 115 126 141 135 125
##  [18] 149 170 170 158 133 114 140 145 150 178 163 172 178 199 199 184 162
##  [35] 146 166 171 180 193 181 183 218 230 242 209 191 172 194 196 196 236
##  [52] 235 229 243 264 272 237 211 180 201 204 188 235 227 234 264 302 293
##  [69] 259 229 203 229 242 233 267 269 270 315 364 347 312 274 237 278 284
##  [86] 277 317 313 318 374 413 405 355 306 271 306 315 301 356 348 355 422
## [103] 465 467 404 347 305 336 340 318 362 348 363 435 491 505 404 359 310
## [120] 337 360 342 406 396 420 472 548 559 463 407 362 405 417 391 419 461
## [137] 472 535 622 606 508 461 390 432
```

```r
exp(as.numeric(forecast_ap1$mean)) # mean/lower/upper is the predicted series
```

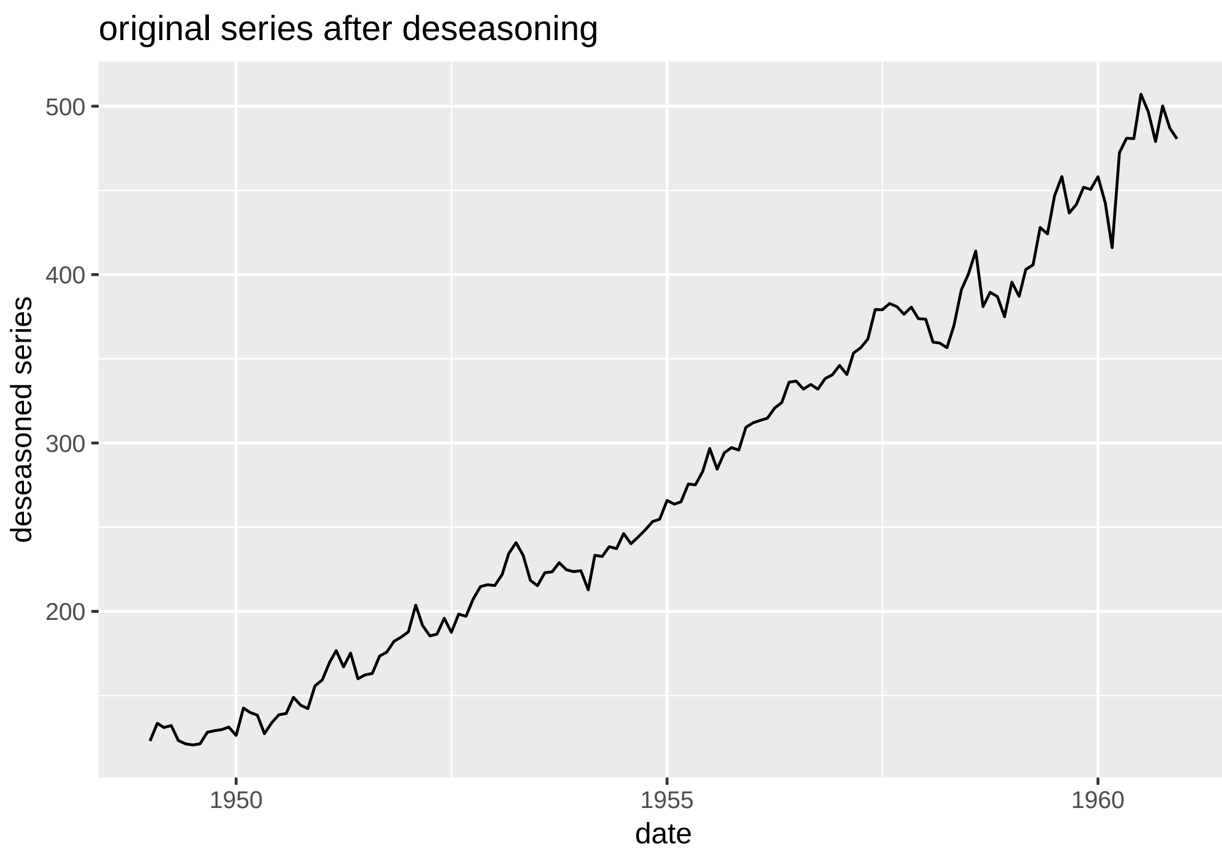
```
##  [1] 450.4224 425.7172 479.0068 492.4045 509.0550 583.3449 670.0108
##  [8] 667.0776 558.1894 497.2078 429.8720 477.2426 495.9301 468.7289
## [15] 527.4025 542.1538 560.4865 642.2823 737.7043 734.4748 614.5852
## [22] 547.4424 473.3034 525.4600 546.0356 516.0862 580.6879 596.9295
## [29] 617.1144 707.1743 812.2371 808.6813 676.6788 602.7524 521.1229
## [36] 578.5491
```

seasonal adjustment

```r
summary(decompose_ap)
```

```
##          Length Class  Mode
## x        144    ts     numeric
## seasonal 144    ts     numeric
## trend    144    ts     numeric
## random   144    ts     numeric
## figure    12    -none- numeric
## type       1    -none- character
```

```r
autoplot(data_ap/decompose_ap$seasonal)+labs(x="date", y="deseasoned series", title = "original series a
```



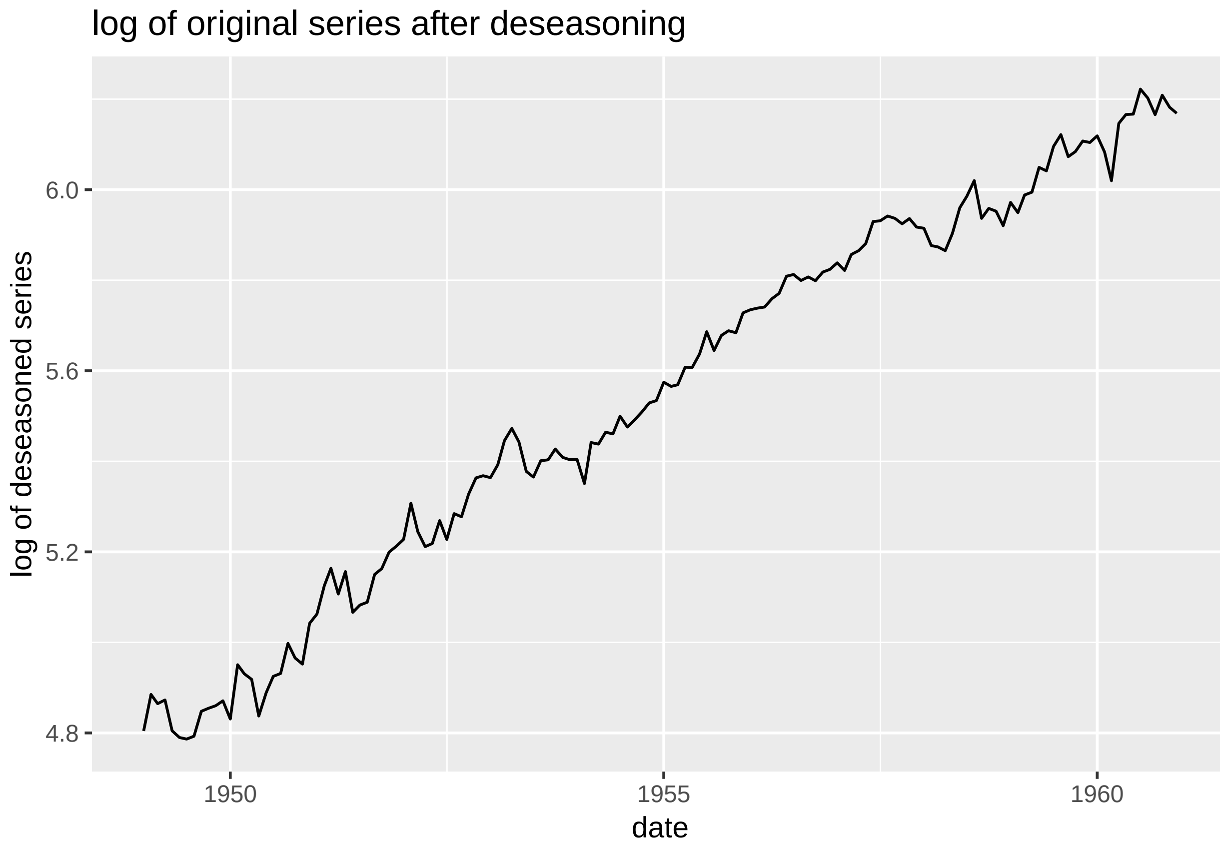
since we choose `multiplicative` in `decompose`, here we need devide by seasonal.

But for logarithm of data, we can use `-`.

```
summary(decompose_ap1)
```

```
##           Length Class  Mode
## x         144    ts     numeric
## seasonal  144    ts     numeric
## trend     144    ts     numeric
## random    144    ts     numeric
## figure     12    -none- numeric
## type        1    -none- character
```

```
autoplot(log(data_ap)-decompose_ap1$seasonal)+labs(x="date", y="log of deseasoned series", title = "log
```


log of original series after deseasoning
log of deseasoned series
6.0
5.6
5.2
4.8
1950
1955
1960
date