# Day 7 - Intro to forecasting strategies

## Introduction

Last week, we explored autocorrelation, and learned how to identify when serial correlation is left in the residual error series. This week, we begin learning traditional time series techniques that may be used to address serial correlation in the residual error series, enabling forecasting techniques.

To guide our discussion, we will use a data set describing the building approvals for each quarter, and the Building Activity Publication lists in Australia, provided by the Australian Bureau of Statistics. The approvals column describes the average number of dwellings approved during each quarter and the activity column describes a measure of the total amount of money spent of creating dwellings each quarter.

```
# packages
library(tidyverse)

# read table from the web
www <- "https://raw.githubusercontent.com/prabeshdhakal/Introductory-Time-Series-with-R-Da
aus_dat <- read.table(www, header = T)

# build ts for each series
ts_app <- ts(
  aus_dat$Approvals,
  start = c(1996, 1),
  freq = 4
)

ts_act <- ts(
  aus_dat$Activity,
  start = c(1996, 1),
  freq = 4
)
```

---

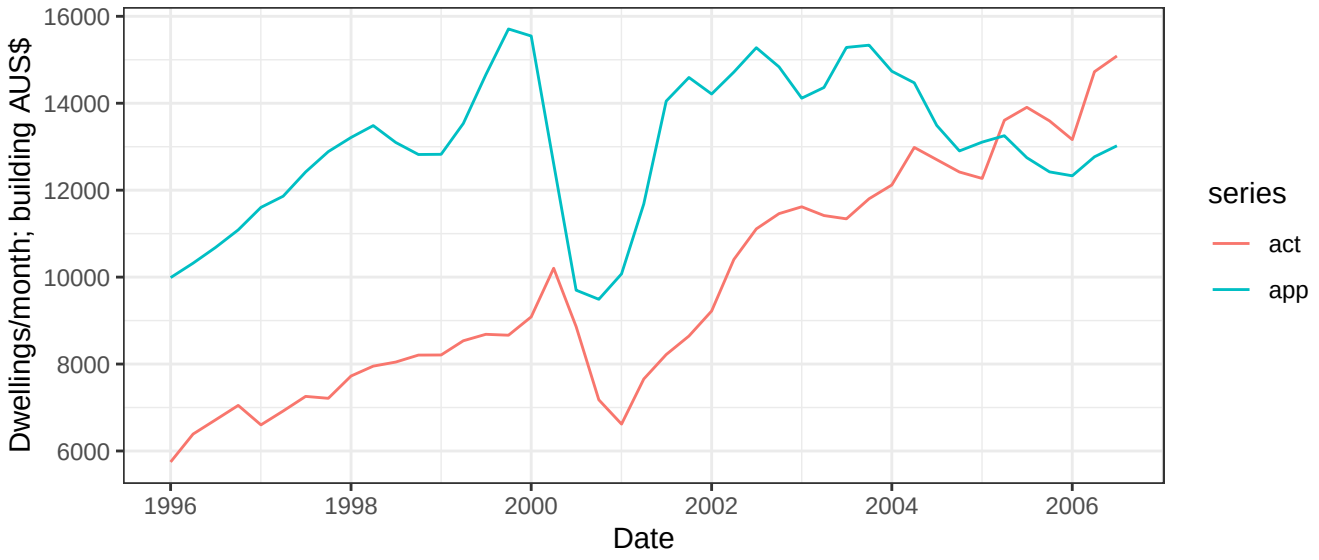
## Leading variables

> **Note**
>
> When forecasting, our goal is to predict some future value $x_{n+k}$ given a history of $\{x_1, x_2, \ldots, x_n\}$ observations up to time $n$. In some simple cases, usually related to production or sales, one time series may preceed another series, providing information about the second series.

```
# could also just run:
# ts.plot(ts_app, ts_act, lty = c(1,3))

# making a nice ggplot
aus <- tibble(
  year = rep(1996:2006, each = 4)[1:43],
  quarter = rep(1:4, 11)[1:43],
  month = rep(c(1, 4, 7, 10), 11)[1:43],
  app = aus_dat$Approvals,
  act = aus_dat$Activity
) %>%
  mutate(dt = ymd(paste0(year, "-", month, "-1")))

aus %>%
  pivot_longer(app:act, names_to = "series", values_to = "val") %>%
  ggplot() +
  geom_line(aes(x = dt, y = val, col = series)) +
  theme_bw() +
  labs(y = "Dwellings/month; building AUS$", x = "Date")
```



**Note**

The _____ (ccvf) describes the autocorrelation between a time series at time $t$ and another time series at time $t + k$.

$$\gamma_k(X, Y) = E\left[(X_{t+k} - \mu_x)(Y_t - \mu_y)\right]$$

The _____ (ccf) is given by:

$$\rho_k(X, Y) = \frac{\gamma_k(X, Y)}{\sigma_x \sigma_y}$$

Sample estimates are obtained in the usual way, with the sample cross-covariance function given by:

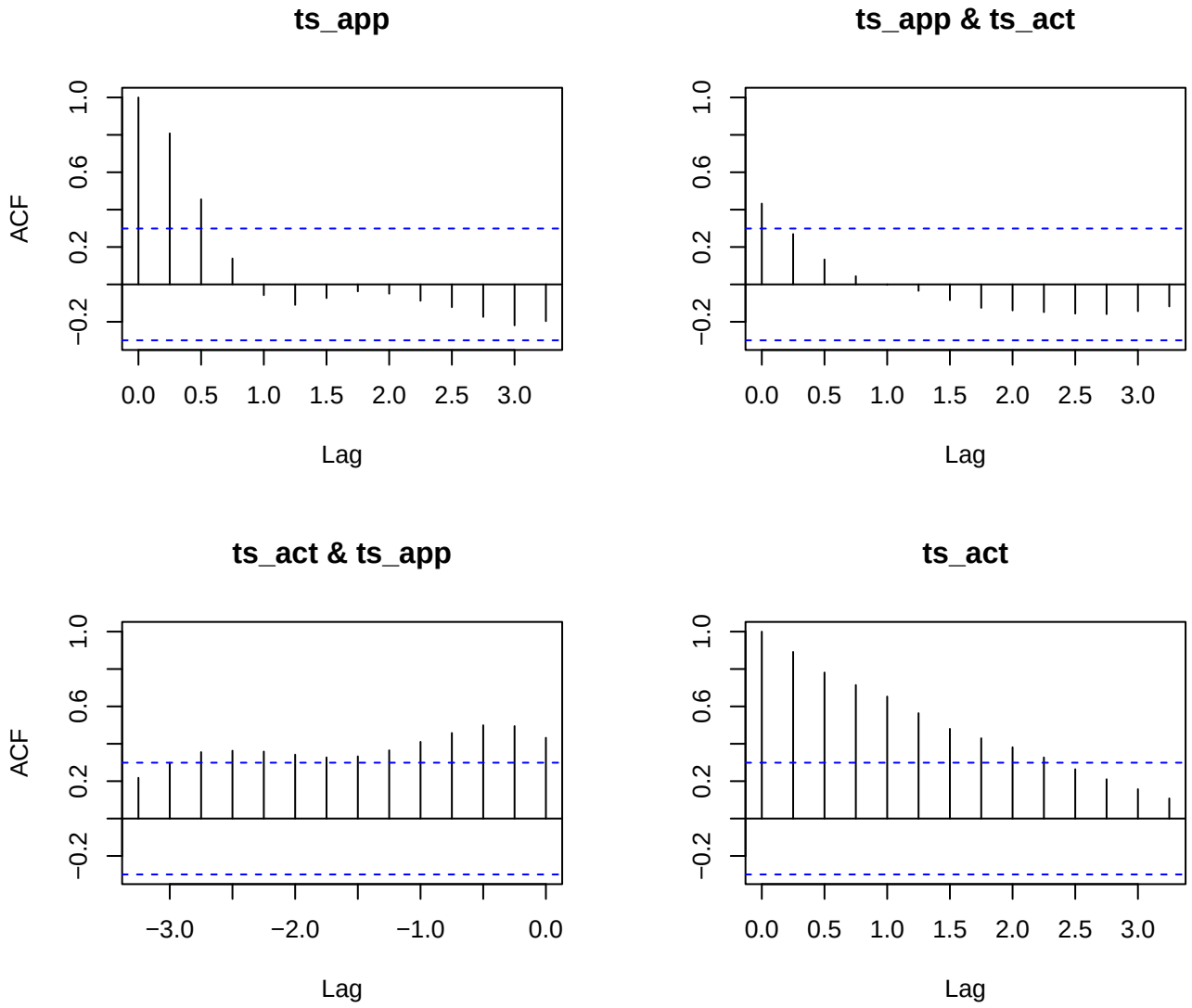
$$c_k(x, y) = \frac{1}{n} \sum_{t=1}^{n-k} (x_{t+k} - \bar{x})(y_t - \bar{y})$$

and the sample cross-correlation function given by:

$$r_k(x, y) = \frac{c_k(x, y)}{\sqrt{c_0(x, x) c_0(y, y)}}$$

The `ts.union` and `acf` functions may be used to visualize the cross-correlation between two time series.

```
acf(ts.union(ts_app, ts_act))
```

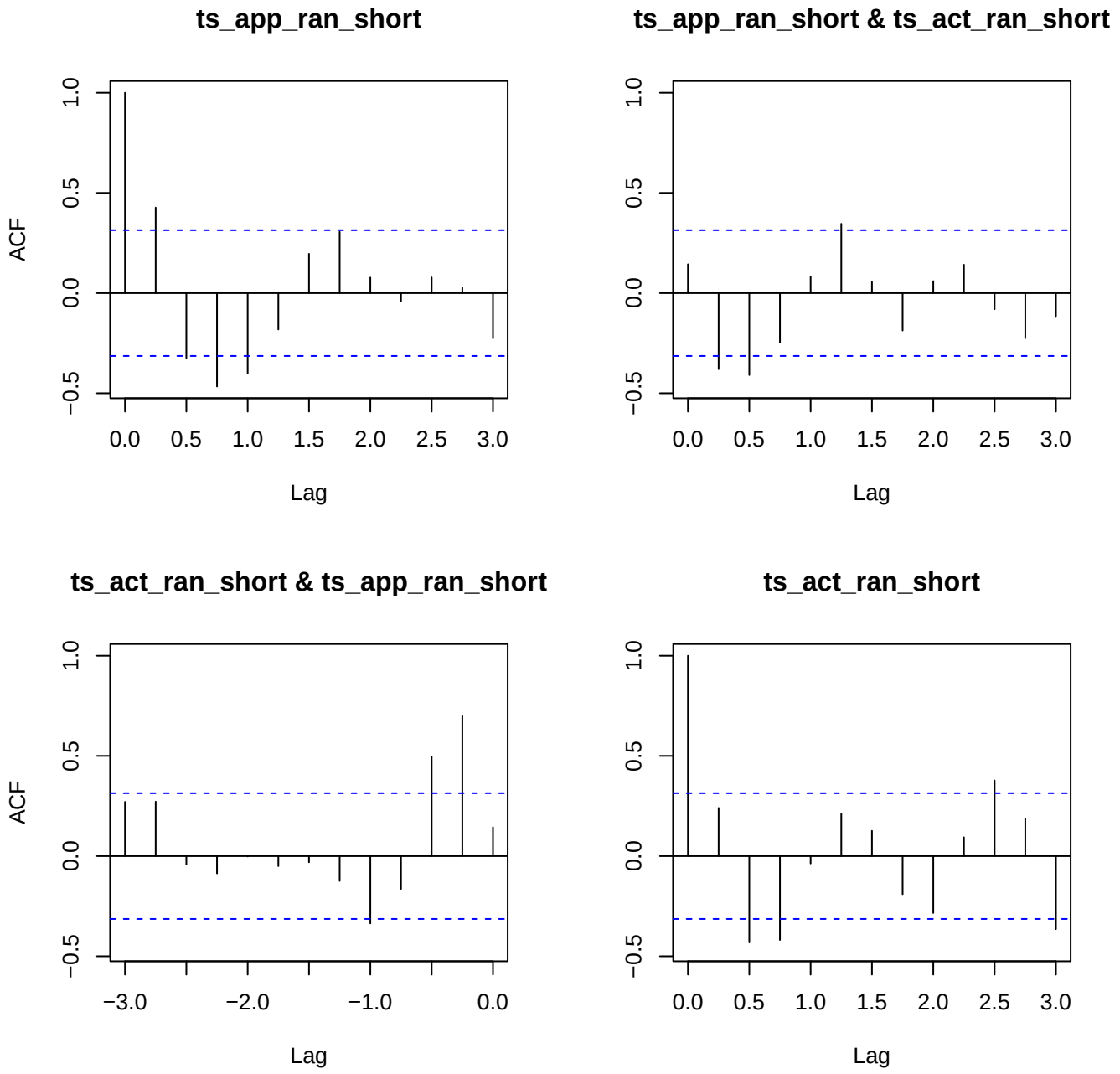



It is possible that the cross-correlation is largely driven by a shared trend or seasonality. We can recreate the cross-correlation plot after first removing the trend and seasonal components using `decompose`.

```
# grab the random components
ts_app_ran <- decompose(ts_app)$random
ts_app_ran_short <- window(
  ts_app_ran,
  start = c(1996, 3), # remove first 2 NAs
  end = c(2006, 1) # remove last 2 NAs
)
ts_act_ran <- decompose(ts_act)$random
ts_act_ran_short <- window(
  ts_act_ran,
```

```
  start = c(1996, 3), # remove first 2 NAs
  end = c(2006, 1) # remove last 2 NAs
)

# cross correlation plot
acf(ts.union(ts_app_ran_short, ts_act_ran_short))
```



## Exponential smoothing

**Note**

If we assume that systematic trend and seasonal effects have been removed from a series, _____ provides a way to model a time series and obtain a forecasts:

$$x_t = \mu_t + w_t$$

where $\mu_t$ and $w_t$ represent the non-stationary mean at time $t$ and independent random deviations with mean 0 and standard deviation $\sigma$, respectively.

In order to fit the exponential smoothing model, we must estimate $\mu_t$. What is a reasonable approach?

**Note**

The estimate of $\mu_t$, denoted $a_t$, is

$$a_t = \alpha x_t + (1 - \alpha) a_{t-1}$$

for $0 < \alpha < 1$. Since we assume that there are no seasonal effects or systematic trends, the forecasting equation is

$$\hat{x}_{n+k|n} = a_n$$

for $n = 1, 2, ...$

In order to obtain $a_t$, what must be estimated?

**Note**

The _____, $e_t$, is defined by:

$$e_t = x_t - \hat{x}_{t|t-1} = x_t - a_{t-1}$$

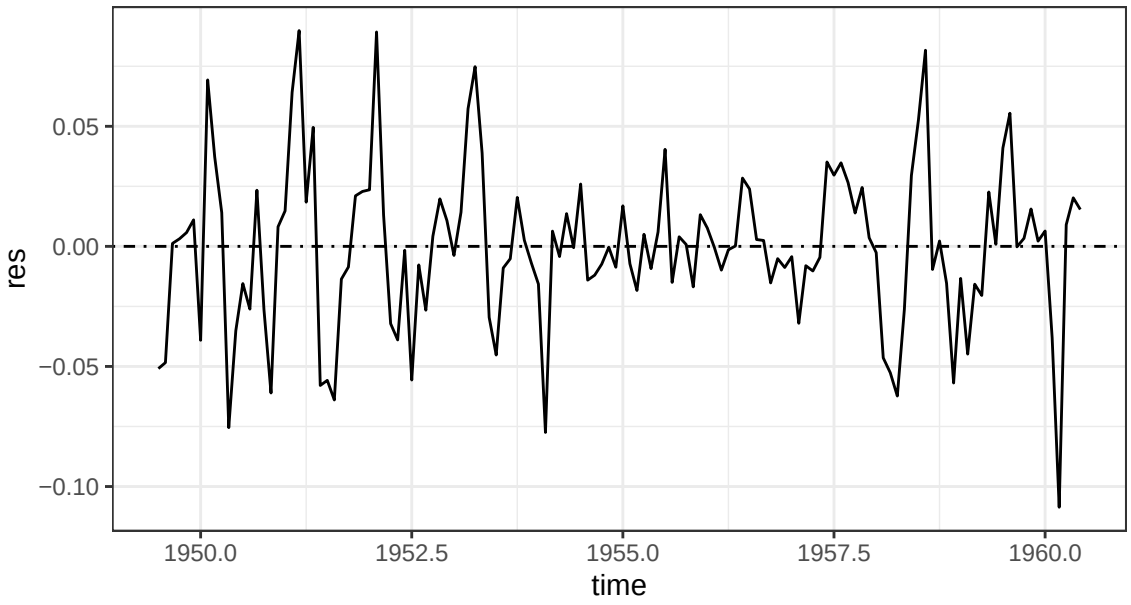
To estimate $\alpha$, R minimizes the sum of the squared $e_t$, assuming that $a_1 = x_1$:

$$SS1PE = \sum_{t=2}^{n} e_t^2 = e_2^2 + e_3^2 + \cdots + e_n^2$$

*Example*: Let us consider applying exponential smoothing to the residual error series from the `AirPassengers` data on the log scale.

```
# demonstrate on Air Passengers (log-scale)
log_ap <- log(AirPassengers)
log_ap_random <- decompose(log_ap)$random %>% na.omit()

# plot
tibble(
  res = c(log_ap_random),
  time = time(log_ap_random)
) %>%
  ggplot() +
  geom_line(aes(x = time, y = res)) +
  theme_bw() +
  geom_hline(aes(yintercept = 0), linetype = "dotdash")
```



```r
# the HoltWinters function is used to fit the model
log_ap_hw1 <- HoltWinters(
  log_ap_random, beta = FALSE, gamma = FALSE
)
log_ap_hw1
```

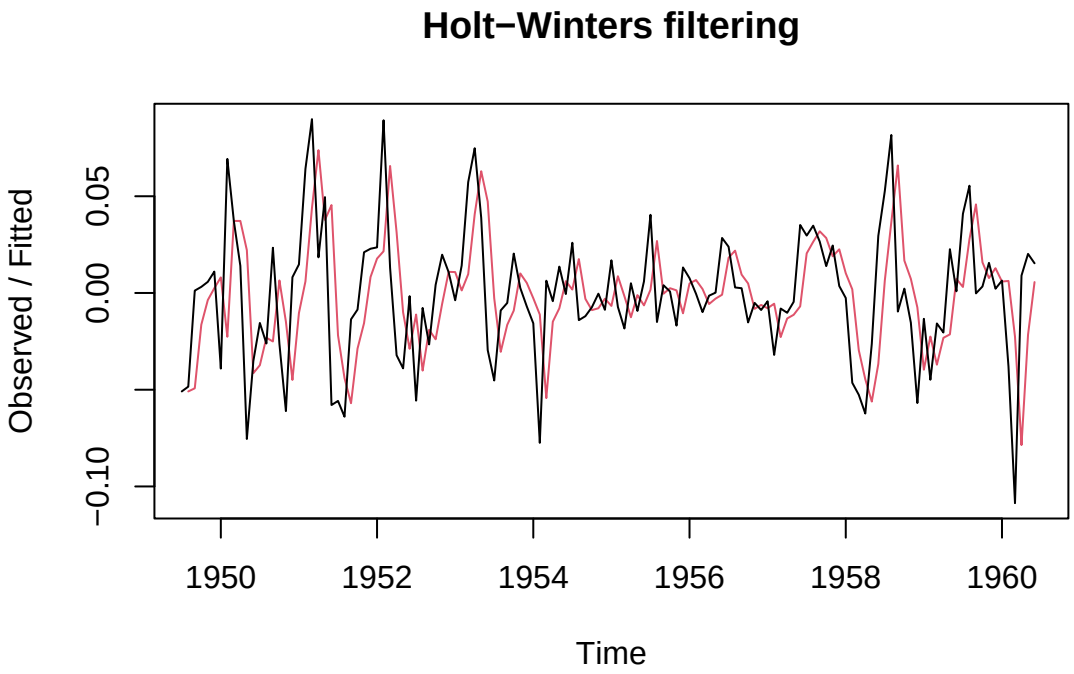
```
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = log_ap_random, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.651189
 beta : FALSE
 gamma: FALSE

Coefficients:
        [,1]
a 0.01196256
```

```r
plot(log_ap_hw1)
```



```r
# adjust the smoothing parameter
log_ap_hw2 <- HoltWinters(
  log_ap_random, alpha = .2, beta = FALSE, gamma = FALSE
)
log_ap_hw2
```

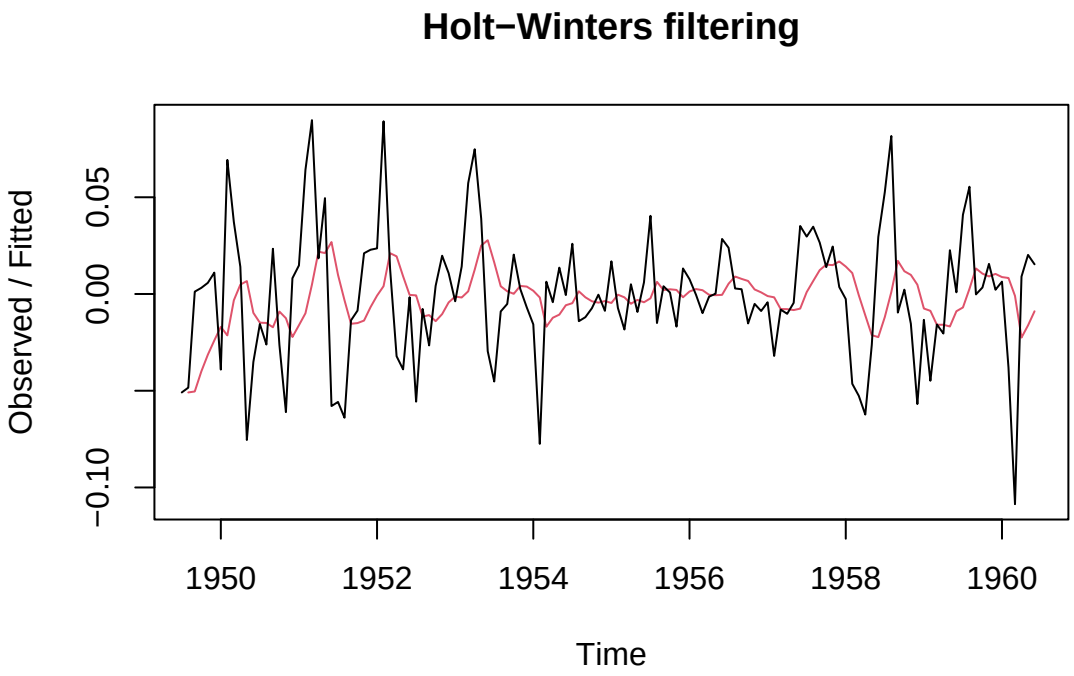
```
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = log_ap_random, alpha = 0.2, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.2
 beta : FALSE
 gamma: FALSE

Coefficients:
          [,1]
a -0.004072477
```

```r
plot(log_ap_hw2)
```

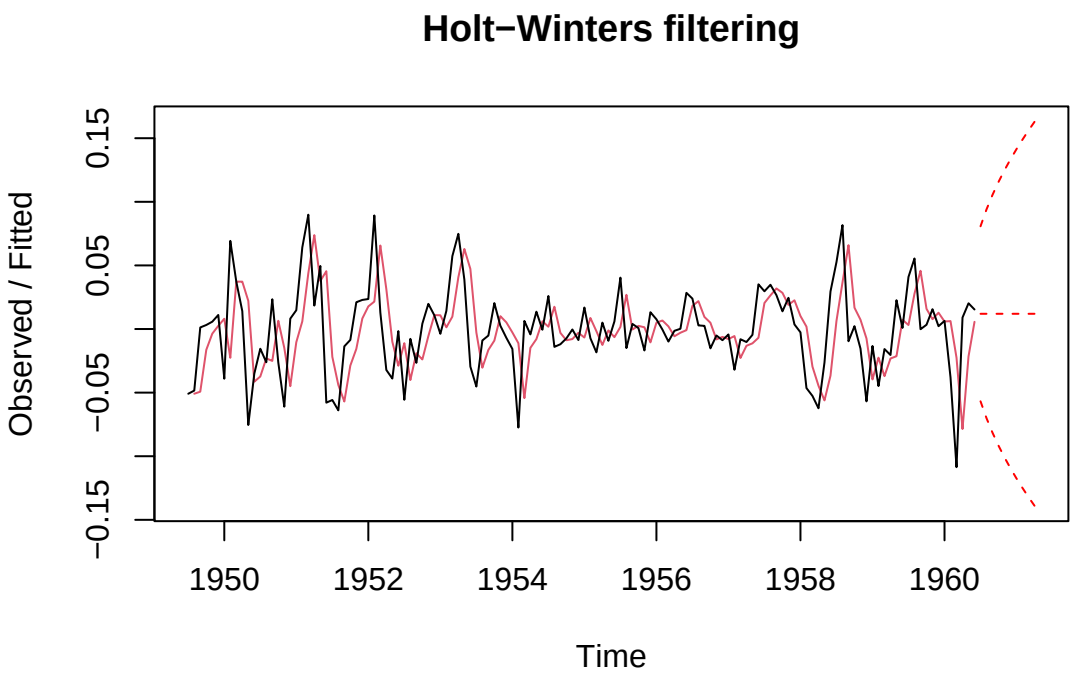


## Exponential smoothing forecasts

```r
# create prediction for 6 time steps ahead
log_ap_hw1_pred <- predict(log_ap_hw1, n.ahead = 10, prediction.interval = TRUE)
log_ap_hw1_pred
```

```
               fit        upr         lwr
Jul 1960 0.01196256 0.08074171 -0.05681659
Aug 1960 0.01196256 0.09403909 -0.07011396
Sep 1960 0.01196256 0.10546412 -0.08153900
Oct 1960 0.01196256 0.11563766 -0.09171254
Nov 1960 0.01196256 0.12489843 -0.10097331
Dec 1960 0.01196256 0.13345534 -0.10953022
Jan 1961 0.01196256 0.14144801 -0.11752289
Feb 1961 0.01196256 0.14897521 -0.12505009
Mar 1961 0.01196256 0.15610989 -0.13218477
Apr 1961 0.01196256 0.16290771 -0.13898258
```

```r
# plot - base R
plot(
  log_ap_hw1,
  xlim = c(
    min(time(log_ap_random)),
    max(time(log_ap_hw1_pred))
  ),
  ylim = c(
    min(log_ap_hw1_pred),
    max(log_ap_hw1_pred)
  )
)
lines(log_ap_hw1_pred[,1], lty = 2, col = "red")
lines(log_ap_hw1_pred[,2], lty = 2, col = "red")
lines(log_ap_hw1_pred[,3], lty = 2, col = "red")
```




```
# plot - ggplot
bind_rows(
  tibble(
    time = time(log_ap_random),
    val = c(log_ap_random)
  ) %>% mutate(type = "obs"),
  bind_rows(
    tibble(
      time = time(log_ap_hw1$fitted[,1]),
      val = c(log_ap_hw1$fitted[,1])
    ),
    tibble(
      time = time(log_ap_hw1_pred),
      val = c(log_ap_hw1_pred[,1]),
      lwr = c(log_ap_hw1_pred[,3]),
      upr = c(log_ap_hw1_pred[,2])
    )
  )  %>% mutate(type = "pred")
) %>%
  ggplot(aes(x = time, y = val, col = type)) +
  geom_line() +
  geom_ribbon(
    aes(ymin = lwr, ymax = upr),
```

```
  linetype = 2,
  alpha = 0.1
) +
theme_bw()
```