R403: Probabilistic and Statistical Computations with R

Topic 14: Linear Regression Models in R

Kaloyan Ganev

2022/2023

Lecture Contents

Introduction

Introduction

- We will not repeat details related to regression analysis
- We only consider the basics of constructing and working with regression models in R
- We stick to linear models

Simple linear regression in R

Simple linear regression in R

- Start again with the Eurostat data on quarterly GDP and its components:

```r
library(eurostat)
library(tidyverse)
library(xts)

search1 <- search_eurostat("GDP", type = "dataset")

data_gdpq <- get_eurostat("namq_10_gdp", time_format = "date")

data_gdpq_bg <- data_gdpq %>%
  filter(geo == "BG",
         unit == "CLV10_MEUR",
         s_adj == "NSA",
         na_item %in% c("B1GQ", "P3", "P5G", "P6", "P7")) %>%
select(time, values, na_item) %>%
spread(na_item, values)  %>%
mutate(time = as.yearqtr(time))
```

- Make an **xts** object from the data frame:

```r
xts1 <- xts(data_gdpq_bg[,2:6],  order.by = data_gdpq_bg$time)
```

Simple linear regression in R (2)

- We will start analysing data by first visualizing them in a scatterplot
- An important note: For the time being we will deliberately allow ourselves making some mistakes related to the properties of time series data (esp. stationarity)
- Take aggregate consumption and GDP; the scatterplot is generated by:

```r
gg1 <- ggplot(xts1) +
  geom_point(aes(x = log(B1GQ), y = log(P3)),
             size = 4, colour = "blue", alpha = 0.5) +
  theme_bw()
```

- The graph suggests a linear relationship
- Run a linear regression using the `lm()` command:

```r
lm1 <- lm(log(P3) ~ log(B1GQ), data = xts1["2000-01/2019-04"])
```

- Check with `mode()` and `class()` the type of the regression object

Simple linear regression in R (3)

- To explore the regression output, the `summary()` command is used:

```
print(lm1)
summary(lm1)
```

- As the model object is a list, we can also access its elements which provide additional useful information
- To get only the coefficients, you can try with:

```
lm1$coefficients
```

- You can also do it with:

```
coef(lm1)
```

- The output is a (named) vector

Simple linear regression in R (4)

- To add the regression line to our scatterplot:

```
gg1 <- gg1 +
    geom_abline(intercept = lm1$coef[1],
                slope = lm1$coef[2],
                size = 1.2,
                colour = "red")
```

- Using base R functionality, the same can be achieved by simply using:

```
abline(lm1)
```

- Fitted values can be accessed with:

```
lm1$fitted # or
fitted(lm1)
```

- We can add the fitted values to the existing **xts** object:

```
xts1$P3_fitted <- c(rep(NA, length(xts1[,2])-length(lm1$fitted)),
                    exp(lm1$fitted))
```

Simple linear regression in R (5)

- In order to plot together the actuals and the fitted values:

```
gg2 <- ggplot(xts1["2000/2019"], aes(x = Index)) +
  geom_line(aes(y = P3, color = "Consumption")) +
  geom_line(aes(y = P3_fitted, color = "Fitted"), lty = 2) +
  xlab("") +
  ylab("") +
  scale_color_manual("", values = c("red", "darkgreen")) +
  ggtitle("Actual and Fitted Consumption") +
  theme_minimal()
```

- Note that the fit indicates the low quality of the regression in many respects (we will see this also in the residual plot)

Simple linear regression in R (6)

- Residuals can be extracted with:

```
lm1$residuals # or:
resid(lm1)
```

- To add them to our **xts** object:

```
xts1$resids <- c( rep(NA, length(xts1[,2])-length(lm1$resid)),
    lm1$resid)
```

- The residual plot:

```
gg3 <- ggplot(xts1["2000/2019"], aes(x = Index)) +
  geom_line(aes(y = resids), color = "red") +
  xlab("") +
  ylab("") +
  ggtitle("Residuals") +
  theme_minimal()
```

Simple linear regression in R (7)

- R provides also a set of diagnostic plots directly accessible through:

```r
plot(lm1)
```

- More info on what the output is about can be found if you search for help on `plot.lm()`
- Some of the output should still be already familiar to you
- What's new maybe is the *scale-location* plot: almost the same as residuals vs. fitted values
- Instead of residuals, however, the square root of the absolute value of standardized residuals is used
- The plot can be considered a rough diagnostic for heteroskedasticity
- Residuals vs. leverage: leverage measures the sensitivity of the model coefficient to individual observations

Simple linear regression in R (8)

- We can perform ANOVA on our simple model to identify sources of variation

```
anova(lm1)
```

- The latter also gives us an F statistic to compare nested models (one with a constant only, and one with GDP as a predictor in our case)
- We can get the confidence intervals for point estimates using:

```
confint(lm1, level = 0.95)
```

- If you are using exactly a 95%-confidence level, then `level = 0.95` can be skipped
- With `deviance()`, `logLik()`, and `AIC()` and `BIC()` you get respectively the sum of squared residuals (RSS), the value of the log-likelihood[1], and AIC and SIC

[1] Note that it is computed under the assumption of normality of disturbances.

Simple linear regression in R (9)

- There are some elements of regression output which we did not discuss
- To get the standard error of residuals $\sqrt{\frac{\sum \varepsilon_i^2}{n-k}}$:

```
sqrt(deviance(lm1)/df.residual(lm1))
```

- The following:

```
df.residual(lm1)
```

will only get you the degrees of freedom $(n-k)$

Simple linear regression in R (10)

- R^2 can be computed in several ways (I am skipping the hopefully familiar formulae):

```
P3_smpl <- as.numeric(log(xts1$P3)["2000/2019"])

var(fitted(lm1))/var(P3_smpl)
cor(P3_smpl, fitted(lm1))**2
1 - var(resid(lm1))/var(P3_smpl)
```

- Otherwise, it can be extracted from regression output in the following way:

```
smr_lm1 <- summary(lm1)
smr_lm1$r.squared
```

- Adjusted $R^2 = 1 - (1 - R^2)\dfrac{n-1}{n-p-1}$:

```
1 - (1 - var(fitted(lm1))/var(P3_smpl)) * (length(P3_smpl)-1)/df.
    residual(lm1)
smr_lm1$adj.r.squared
```

Simple linear regression in R (11)

- To access coefficient estimates, standard errors, t-values, and p-values, you can use matrix subsetting operations
- For example, to get the column of standard errors:

```
coef(summary(lm1))[,2]
```

- To get the t-statistic for consumption:

```
coef(summary(lm1))[2,3]
```

- etc.

Simple linear regression in R (12)

- The t-statistic is simply the ratio of the coefficient estimate and its standard error: $t = \dfrac{\widehat{\beta}}{se(\widehat{\beta})}$
- Thus, for example:

```
t1 <- coef(summary(lm1))[1,1]/coef(summary(lm1))[1,2]
```

- The corresponding p-value is obtained through:

```
p1 <- 2*(1 - pt(t1, df = df.residual(lm1)))
```

Simple linear regression in R (13)

- Regression through the origin:

```
lm1a <- lm(log(P3) ˜ 0 + log(B1GQ), data = xts1["2000/2019"])
summary(lm1a)
```

- Regression on a constant

```
lm1b <- lm(log(P3) ˜ 1, data = xts1["2000/2019"])
summary(lm1b)
```

Simple linear regression in R (14)

- `predict()` allows to forecast using the fitted model when new data on the independent variable are entered
- As an example, run the simple linear regression again but until the end of 2017 only:

```r
lm1c <- lm(log(P3) ~ log(B1GQ), data = xts1["2000-01/2017-04"])
summary(lm1c)
```

- Generate the prediction for 2018-2019 and then merge it to the **xts** dataset:

```r
fc_lm1c <- exp(predict(lm1c, newdata = xts1["2018-01/2019-04"]))
xts1$fc_lm1c <- c(rep(NA, length(xts1$P3)-length(fc_lm1c)), fc_
    lm1c)
```

- This is only for demonstration as it is quite tedious
- There are specialized packages for time series analysis that automate a great part of the work

Simple linear regression in R (15)

- A plot of actual, fitted, and residuals can be generated with:

```
ggplot(xts1["2000/2019"], aes(x = Index)) +
  scale_x_continuous() +
  geom_line(aes(y = P3, color = "Consumption"))  +
  geom_line(aes(y = P3_fitted, color = "Fitted"), lty = 2)  +
  geom_line(aes(y = fc_lm1c, color = "Forecast"), size = 1.2)  +
  xlab("") +
  ylab("") +
  scale_color_manual("", values = c("red", "darkblue", "darkgreen
      ")) +
  ggtitle("Actual, fitted, and forecast consumption") +
  theme_minimal()
```

Multivariate linear regression in R

Multivariate linear regression in R

- We will run a regression with three predictors (not so meaningful, just for illustration):

```
lm2 <- lm(P7 ~ P3 + P5G + P6, data = xts1)
summary(lm2)
```

- Add fitted values to **xts** dataset:

```
xts1$P7_fitted <- fitted(lm2)
```

- ...and plot the results:

```
ggplot(xts1["2000/2019"], aes(x = Index)) +
  scale_x_continuous() +
  geom_line(aes(y = P7, color = "Imports"))  +
  geom_line(aes(y = P7_fitted, color = "Fitted"), lty = 2)  +
  xlab("") +
  ylab("") +
  scale_color_manual("", values = c("red", "darkblue")) +
  ggtitle("Actual and fitted imports") +
  theme_minimal()
```

Residual Tests

Normality tests: The Kolmogorov-Smirnov test

- The test is designed to compare two univariate continuous distributions
- It can be used to compare the distribution of an empirical variable with a pre-specified distribution, in particular the normal one
- Essence: Measures the **maximum** distance between the empirical distribution function ($F_n(x)$) and the cdf of the reference distribution ($F(x)$)

$$D_n = \sup_x |F_n(x) - F(x)|$$

- The null hypothesis is that the data follows the reference distribution

Normality tests: The Kolmogorov-Smirnov test (2)

- In R, the test is implemented with the `ks.test()` function available from the **stats** package
- Using the residuals available for the object `xts1`:

```r
ks.test(xts1$resids, "pnorm", mean = mean(xts1$resids, na.rm = T)
    , sd = sd(xts1$resids, na.rm = T))
```

- We will also plot the empirical vs. the theoretical cumulative distribution:

```r
ggplot(xts1["2000/2019"]) +
  stat_ecdf(aes(x = resids), geom = "point") +
  stat_function(fun = pnorm,
                args = list(mean = mean(xts1$resids, na.rm = T),
                            sd = sd(xts1$resids, na.rm = T)),
                color = "red") +
  xlab("") +
  ylab("") +
  theme_minimal()
```

Normality tests: The Jarque-Bera test

- Checks whether the sample data is characterized with kurtosis and skewness equal to the corresponding parameters of the normal distribution
- The test statistic is:

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right)$$

- The null hypothesis is that the sample is normally distributed
- If the calculated value of the statistic exceeds the critical value, the hypothesis is rejected
- R example:

```
library(moments)
skewness(xts1$resids, na.rm = T)
kurtosis(xts1$resids, na.rm = T)

resids_vec <- as.vector(xts1$resids["2000/2019"])
jarque.test(resids_vec)
```

Normality tests: Other

- Lilliefors test (some correction of K-S): **nortest** package
- Shapiro-Wilk test: `shapiro.wilk()`
- Cramer-von Mises test: **nortest** package
- Anderson-Darling test: **nortest** package
- D'Agostino skewness test: **moments** package
- D'Agostino-Pearson omnibus test: **fBasics** package
- Anscombe-Glynn test: **moments** package
- Mardia test (Multivariate normal): **MVN** package
- Henze-Zirkler (Multivariate normal): **MVN** package
- Doornik-Hansen (Multivariate normal): **normwhn.test** package
- etc.

Autocorrelation tests

- Box–Pierce or Ljung–Box test statistic (from the **stats** package):

```
Box.test(xts1$resids, lag = 2, type = "Box-Pierce", fitdf = 1)
Box.test(xts1$resids, lag = 2, type = "Ljung-Box", fitdf = 1)
```

- Durbin-Watson:

```
library(lmtest)
dwtest(lm1)
```

- Breusch-Godfrey test for higher-order autocorrelation (also requires **lmtest**):

```
bgtest(lm1, order = 4)
```

Heteroskedasticity tests

- Breusch-Pagan test

```
bptest(lm1)
```

- White test:

```
bptest(lm1, ~ log(B1GQ) + I(log(B1GQ)^2), data = xts1)
```

- Goldfeld-Quandt test

```
gqtest(lm1)
```

- More time will be spent on those in the forthcoming courses