Linear regression for time series

Estimation of linear regression using the asymptotic framework

In this short note, we apply the asymptotic framework of the linear regression to time series data. We first show how to perform statistical tests robust to heteroskedasticity. Then, we use financial data to see if the assumptions of the asymptotic framework hold. Our dependent variable is called 'HFI' and denotes a weighted index of all the Hedge fund returns of the credit Suisse database. We shall decompose the general HF strategy into 3 potential risks that are the financial market (approximated by the excess returns of the S&P 500), the SMB (small minus big anomaly) and the default risk (DEF).

In this exercise, we shall do the following:

1. Simulate data and test the finite as well as the asymptotic linear regression framework
2. Apply the asymptotic framework to empirical data

We start by creating an R function, called *OLS_computation*, that computes many relevant statistics of the linear regression framework. In particular, given the dependent and the explanatory variables, we shall compute the OLS estimates, the coefficient of determination, the t- and Fisher- tests and the tests robust to heteroskedasticity.

```
OLS_computation = function(y,X){
  T = length(y)
  K = dim(X)[2]
  if(dim(X)[1]!=length(y)){stop("Explanatory variables should have the same length as the dependent vari
  if(sum(diag(var(X))==0)==0){
    ## We add an intercept to the explanatory variable
    X_old = X
    X = array(1,dim=c(T,K+1))
    X[,2:(K+1)] = X_old
    K = dim(X)[2]
  }

  inv_X = solve(t(X)%*%X)
  beta_ols = inv_X%*%(t(X)%*%y)
  epsi = y - X%*%beta_ols

  SSR_U = sum(epsi^2)
  SSR_R = sum((y-mean(y))^2)
  R_sq = 1-SSR_U/SSR_R

  ## Finite statistical results or asymptotic with i.i.d. r.v.
  sigma_hat = sum(epsi^2)/(T-length(beta_ols))
  beta_var = sigma_hat*inv_X
  t_stat = beta_ols/sqrt(diag(beta_var))
  p_val = 2*(1-pnorm(abs(t_stat)))
  df = data.frame(cbind(beta_ols,sqrt(diag(beta_var)),t_stat,p_val))
  colnames(df) =c("Estimate","Std.Error","t_value","Pr(>|t|)")
```

```
  ## Linear constraint with asymptotic i.i.d. r.v.
  df1 = length(beta_ols)-1
  df2 = T-length(beta_ols)
  F_test = (df2/df1)*((SSR_R-SSR_U)/SSR_U)
  F_p_val = 1-pf(F_test,df1,df2)
  df_F = data.frame(cbind(F_test,F_p_val))
  colnames(df_F) =c("F_stat","P value")


  ## Asymptotic framework robust to heteroskedasticity
  robust_S = array(0,c(K,K))
  for(t in 1:T){
    robust_S = robust_S + epsi[t]^2*(as.matrix(X[t,])%*%X[t,])/T
  }
  beta_var_robust = ((T*inv_X)%*%robust_S%*%(T*inv_X))/T ## Divide by T to get the variance of (beta_ha
  t_stat_robust = beta_ols/sqrt(diag(beta_var_robust))
  p_val_robust = 2*(1-pnorm(abs(t_stat_robust)))
  df_robust = data.frame(cbind(beta_ols,sqrt(diag(beta_var_robust)),t_stat_robust,p_val_robust))
  colnames(df_robust) =c("Estimate","Std.Error (het)","t_value (het)","Pr(>|t|) (het)")



  ## test linear constraints robust to heteroskedasticity
  R = diag(K)
  R = R[2:K,1:K]
  stat_wald = T*t(R%*%beta_ols)%*%solve(R%*%(T*beta_var_robust)%*%t(R))%*%(R%*%beta_ols)
  wald_p_val = 1-pchisq(stat_wald,K-1)
  df_wald = data.frame(cbind(stat_wald,wald_p_val))
  colnames(df_wald) =c("Wald_stat","P value")

  result = list("OLS_results" = df,"OLS_results_robust_het" = df_robust, "F_test" = df_F,"R_sq" = R_sq,
  return(result)
}
```

We now code a function, called *autocorrelation_BG*, that compute the Breush-Godfrey test (i.e., an autocorrelation test on the residuals). The function allows to choose the number of lags of the residuals to be used in tests.

```
autocorrelation_BG = function(y,X,p=4){
  T = length(y)
  K = dim(X)[2]
  if(dim(X)[1]!=length(y)){stop("Explanatory variables should have the same length as the dependent vari

  inv_X = solve(t(X)%*%X)
  beta_ols = inv_X%*%(t(X)%*%y)
  epsi = y - X%*%beta_ols

  ## Autocorrelation test with p lags
  X_eps = array(0,c(T-p,p))
  for(i in 1:p){
    X_eps[,i] = epsi[(i):(T-p+(i-1))]
  }
  y_eps = epsi[(p+1):T]
  X_all = cbind(X_eps,X[(p+1):T])
```

```
  inv_X = solve(t(X_all)%*%X_all)
  beta_ols = inv_X%*%(t(X_all)%*%y_eps)
  eta = y_eps - X_all%*%beta_ols

  K = length(beta_ols)
  T = length(y_eps)
  robust_S = array(0,c(K,K))
  for(t in 1:T){
    robust_S = robust_S + eta[t]^2*(as.matrix(X_all[t,])%*%X_all[t,])/T
  }
  beta_var_robust = ((T*inv_X)%*%robust_S%*%(T*inv_X))/T ## Divide by T to get the variance of (beta_ha

  ## test linear constraints robust to heteroskedasticity
  R = diag(K)
  R = R[1:p,1:K]
  stat_wald = T*t(R%*%beta_ols)%*%solve(R%*%(T*beta_var_robust)%*%t(R))%*%(R%*%beta_ols)
  wald_p_val = 1-pchisq(stat_wald,p)
  df_wald = data.frame(cbind(stat_wald,qchisq(0.95,p),wald_p_val))
  colnames(df_wald) =c("Wald_stat","Critical value (95%)","P value")
  results = list("residuals" = epsi,"Breush_Godfrey_test"=df_wald)
  return(results)
}
```

We can now test the theory with simulated data. We will then use financial data. We first simulate a regression with error terms that are i.i.d. but not Normally distributed. In such a case, the Gaussian assumption is violated but the asymptotic framework with homoskedastic conditional variance applies.

```
T = 1000
X = array(rt(4*T,10),c(T,4)) ## Simulation from a student distribution
X[,1] = 1
beta_true = c(2,0,-0.15,-1.2)
y = X%*%beta_true + rt(T,5) ## The error term is i.i.d. student(5)
reg_iid = OLS_computation(y,X)
(reg_iid)
```

```
## $OLS_results
##      Estimate  Std.Error     t_value     Pr(>|t|)
## 1  2.00821210 0.04160649  48.2667990 0.000000e+00
## 2  0.01176895 0.03819554   0.3081237 7.579882e-01
## 3 -0.20177698 0.03734844  -5.4025541 6.569862e-08
## 4 -1.17421742 0.03949918 -29.7276386 0.000000e+00
##
## $OLS_results_robust_het
##      Estimate Std.Error (het) t_value (het) Pr(>|t|) (het)
## 1  2.00821210      0.04140369    48.5032152   0.000000e+00
## 2  0.01176895      0.03799370     0.3097606   7.567430e-01
## 3 -0.20177698      0.03343946    -6.0340973   1.598538e-09
## 4 -1.17421742      0.03692065   -31.8038149   0.000000e+00
##
## $F_test
##   F_stat P value
## 1 308.58       0
##
```

```
## $R_sq
## [1] 0.4817197
##
## [[5]]
## [1] "Wald_test"
##
## [[6]]
##   Wald_stat P value
## 1  1115.633       0
```

```
ac_iid = autocorrelation_BG(y,X)
(ac_iid$Breush_Godfrey_test)
```

```
##   Wald_stat Critical value (95%)   P value
## 1  4.391524             9.487729 0.3556044
```

We observe that the OLS estimates seem consistent (their values are close to the true ones). In addition, the robust standard deviations are close to the ones assuming conditional homoskedasticity (i.e. the standard values, see *OLS_results*). In the next example, we include heteroskedasticity.

```
T = 1000
X = array(rt(4*T,10),c(T,4)) ## Simulation from a student distribution
X[,1] = 1
beta_true = c(2,0,-0.15,-1.2)

epsi = rt(T,5)
for(t in 2:T){
  epsi[t] = X[t,2]*rt(1,5)
}
y = X%*%beta_true + epsi ## The error term is identically distributed student(5) with var(e_t)= x_2^2*d
reg_het = OLS_computation(y,X)
(reg_het)
```

```
## $OLS_results
##      Estimate  Std.Error    t_value     Pr(>|t|)
## 1  2.06114469 0.04256227  48.426571 0.000000e+00
## 2 -0.06871653 0.03858097  -1.781099 7.489625e-02
## 3 -0.14582146 0.03747128  -3.891553 9.960482e-05
## 4 -1.22806685 0.03621634 -33.909196 0.000000e+00
##
## $OLS_results_robust_het
##      Estimate Std.Error (het) t_value (het) Pr(>|t|) (het)
## 1  2.06114469      0.04235844    48.6595993   0.000000e+00
## 2 -0.06871653      0.07169647    -0.9584368   3.378426e-01
## 3 -0.14582146      0.03155407    -4.6213210   3.813043e-06
## 4 -1.22806685      0.04183379   -29.3558596   0.000000e+00
##
## $F_test
##     F_stat P value
## 1 388.2888       0
##
## $R_sq
## [1] 0.5390738
##
## [[5]]
## [1] "Wald_test"
```

```
##
## [[6]]
##   Wald_stat P value
## 1  880.6064       0
```

The robust standard deviations are now different from those obtained from standard results. Statistical tests can therefore be misleading if they are based on the assumption of conditional homoskedasticity. For example, note that the second parameter is significant at a level of 10% using the non-robust standard deviations.

We end these simulation approaches by simulating a linear regression with autocorrelated error terms. In such a case, the error term is not anymore a martingale difference sequence and so, we cannot invoke the CLT for statistical tests.

```
T = 1000
X = array(rt(4*T,10),c(T,4))
X[,1] = 1
beta_true = c(2,0,-0.15,-1.2)

epsi = rt(T,5)
for(t in 2:T){
  epsi[t] = 0.95*epsi[t-1] + rt(1,5)
}
y = X%*%beta_true + epsi

reg_ac = OLS_computation(y,X)
(reg_ac)
```

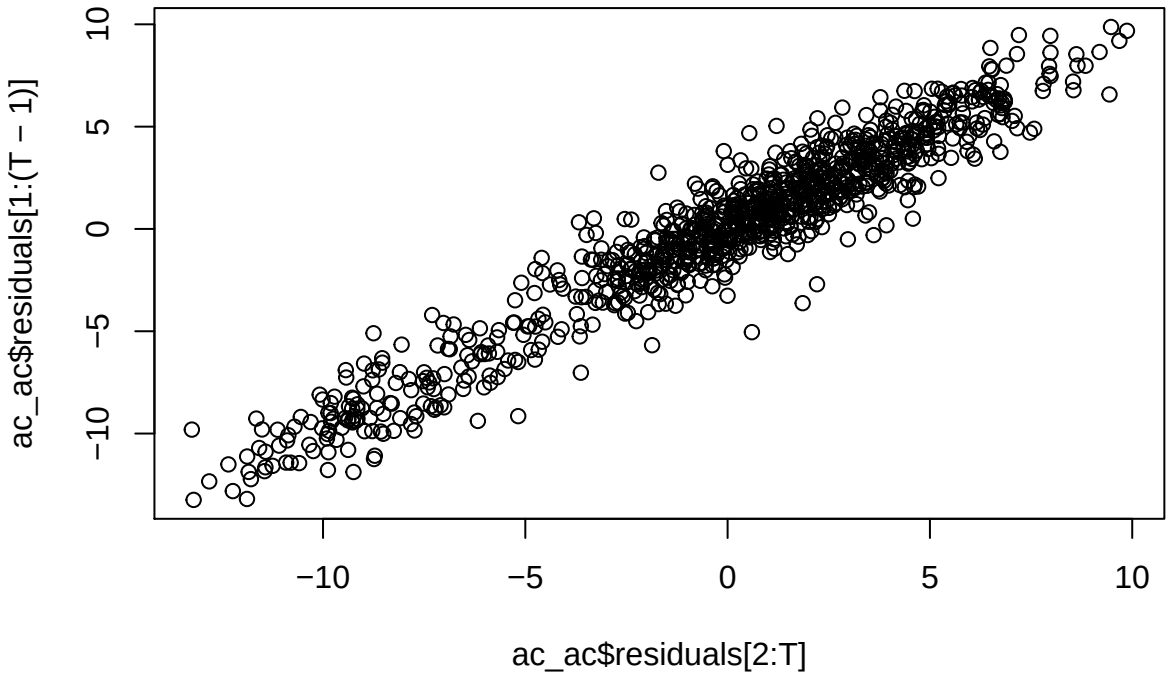
```
## $OLS_results
##      Estimate Std.Error    t_value  Pr(>|t|)
## 1  1.62782397 0.1430443 11.3798582 0.0000000
## 2  0.17934914 0.1275582  1.4060186 0.1597186
## 3 -0.04644714 0.1230696 -0.3774053 0.7058724
## 4 -1.15448328 0.1205267 -9.5786524 0.0000000
##
## $OLS_results_robust_het
##      Estimate Std.Error (het) t_value (het) Pr(>|t|) (het)
## 1  1.62782397       0.1425895    11.4161586      0.0000000
## 2  0.17934914       0.1308047     1.3711219      0.1703369
## 3 -0.04644714       0.1227255    -0.3784638      0.7050861
## 4 -1.15448328       0.1204179    -9.5873034      0.0000000
##
## $F_test
##     F_stat P value
## 1 31.22053       0
##
## $R_sq
## [1] 0.08595474
##
## [[5]]
## [1] "Wald_test"
##
## [[6]]
##   Wald_stat P value
## 1  92.49319       0
```

```
ac_ac = autocorrelation_BG(y,X)
(ac_ac$Breush_Godfrey_test)
```

```
##   Wald_stat Critical value (95%) P value
## 1  11949.43             9.487729       0
```

Interestingly, we reject the Null of no autocorrelation using the Breush-Godfrey test. So, we should try to find another regression to eliminate these autocorrelations. A simple heuristic test of autocorrelation consists in plotting the residuals with respect to its past value (to test the autocorrelation of order 1).

```
T = length(ac_ac$residuals)
plot(ac_ac$residuals[2:T],ac_ac$residuals[1:(T-1)])
```



Eventually, we apply our linear regression framework to time series data.

```
url = "https://raw.githubusercontent.com/adufays/GDP_expectancy/main/DATA_HF.csv"
R_DATA <- read.csv(url)
X = cbind(R_DATA$Intercept,R_DATA$Mkt,R_DATA$SMB,R_DATA$DEF)
y = R_DATA$HF.Index
date = as.Date(R_DATA$date,'%d-%m-%Y')

reg = OLS_computation(y,X)
(reg)
```

```
## $OLS_results
##      Estimate  Std.Error   t_value     Pr(>|t|)
## 1  0.52441972 0.09429910  5.561238 2.678681e-08
## 2  0.23440136 0.02231492 10.504243 0.000000e+00
```

```
## 3  0.09218527 0.02996601  3.076328 2.095672e-03
## 4 -2.32668407 0.52620735 -4.421611 9.796784e-06
##
## $OLS_results_robust_het
##      Estimate Std.Error (het) t_value (het) Pr(>|t|) (het)
## 1  0.52441972      0.09025351      5.810519   6.227953e-09
## 2  0.23440136      0.02838746      8.257214   2.220446e-16
## 3  0.09218527      0.04131862      2.231083   2.567561e-02
## 4 -2.32668407      0.62530905     -3.720855   1.985497e-04
##
## $F_test
##     F_stat P value
## 1 69.39818       0
##
## $R_sq
## [1] 0.4437276
##
## [[5]]
## [1] "Wald_test"
##
## [[6]]
##   Wald_stat P value
## 1  152.4481       0
```

```
ac = autocorrelation_BG(y,X)
(ac$Breush_Godfrey_test)
```

```
##   Wald_stat Critical value (95%)   P value
## 1  2.507651             9.487729 0.6432662
```

Importantly, we do not reject the Null hypothesis of no autocorrelation. We also see that some heteroskedasticity dynamics are likely to exist in the data since the two estimates of the standard deviations are quite different (even though the conclusions of the tests remain identical at 95%). We can safely interpret the results of the linear regression (if we believe in the other assumptions such as stationarity, ergodicity and contemporaneous exogeneity).