Honors_thesis_1020_Main results

Yueheng Hu

2022-10-21

```
setwd("/Users/yuehenghu/Desktop/RP/RP")
```

Pre-estimation

Fill in the missing population data:

```
###raw data########
population <- readxl::read_excel("/Users/yuehenghu/Desktop/RP/RP/population.xlsx")
province <- population$Region[1:30]
pop1 <- readxl::read_excel("/Users/yuehenghu/Desktop/RP/RP/population.xlsx",range = "Sheet2!A1:T31")
pop1 <- t(pop1)
province -> colnames(pop1)

as_tibble(pop1) %>%
  mutate(Year =rownames(pop1)) %>%
  select(Year, c(province))-> pop_df
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(province)‘ instead of ‘province‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
pop_df %>%
  mutate(Year = paste0(Year, "-01-01")) %>%
  mutate(Year = ymd(Year)) %>%
  as_tsibble(index = Year)  -> pop_ts
pop_ts
```

```
## # A tsibble: 20 x 31 [1D]
##    Year       Beijing Tianjin Hebei Shanxi Inner~1 Liaon~2 Jilin Heilo~3 Shang~4
##    <date>       <dbl>   <dbl> <dbl>  <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1 2000-01-01    1364    1001  6674   3247    2372    4184  2682    3807    1609
##  2 2001-01-01    1385    1004  6699   3272    2381    4194  2691    3811    1668
##  3 2002-01-01    1423    1007  6735   3294    2384    4203  2699    3813    1713
##  4 2003-01-01    1456    1011  6769   3314    2386    4210  2704    3815    1766
##  5 2004-01-01    1493    1024  6809   3335    2393    4217  2709    3817    1835
##  6 2005-01-01    1538    1043  6851   3355    2403    4221  2716    3820    1890
```

```
##  7 2006-01-01    1601    1075  6898    3375    2415    4271  2723    3823    1964
##  8 2007-01-01    1676    1115  6943    3393    2429    4298  2730    3824    2064
##  9 2008-01-01    1771    1176  6989    3411    2444    4315  2734    3825    2141
## 10 2009-01-01    1860    1228  7034    3427    2458    4341  2740    3826    2210
## 11 2010-01-01    1962    1299  7194    3574    2472    4375  2747    3833    2303
## 12 2011-01-01    2024    1341  7232    3562    2470    4379  2725    3782    2356
## 13 2012-01-01    2078    1378  7262    3548    2464    4375  2698    3724    2399
## 14 2013-01-01    2125    1410  7288    3535    2455    4365  2668    3666    2448
## 15 2014-01-01    2171    1429  7323    3528    2449    4358  2642    3608    2467
## 16 2015-01-01    2188    1439  7345    3519    2440    4338  2613    3529    2458
## 17 2016-01-01    2195    1443  7375    3514    2436    4327  2567    3463    2467
## 18 2017-01-01    2194    1410  7409    3510    2433    4312  2526    3399    2466
## 19 2018-01-01    2192    1383  7426    3502    2422    4291  2484    3327    2475
## 20 2019-01-01    2190    1385  7447    3497    2415    4277  2448    3255    2481
## # ... with 21 more variables: Jiangsu <dbl>, Zhejiang <dbl>, Anhui <dbl>,
## #   Fujian <dbl>, Jiangxi <dbl>, Shandong <dbl>, Henan <dbl>, Hubei <dbl>,
## #   Hunan <dbl>, Guangdong <dbl>, Guangxi <dbl>, Hainan <dbl>, Chongqing <dbl>,
## #   Sichuan <dbl>, Guizhou <dbl>, Yunnan <dbl>, Shaanxi <dbl>, Gansu <dbl>,
## #   Qinghai <dbl>, Ningxia <dbl>, Xinjiang <dbl>, and abbreviated variable
## #   names 1: ‘Inner Mongolia‘, 2: Liaoning, 3: Heilongjiang, 4: Shanghai
```

```
# try auto ARIMA forecasts
foreresults <- matrix(0,30,3)
for (i in c(1:30)) {
  pop2_i<- ts(pop1[,i])
  #kpss.test(pop2_i, null = c("Trend"), lshort = TRUE)
  model <- pop2_i %>%
    auto.arima() %>%
    forecast::forecast(h=3)
  foreresults[i,] <- model$mean
}
foreresults
```

```
##            [,1]      [,2]      [,3]
##  [1,] 1344.3582 1325.9867 1308.8035
##  [2,]  998.4354  996.2430  994.3688
##  [3,] 6633.3158 6592.6316 6551.9474
##  [4,] 3233.8421 3220.6842 3207.5263
##  [5,] 2363.0000 2354.0000 2345.0000
##  [6,] 4174.0000 4164.0000 4154.0000
##  [7,] 2673.0000 2664.0000 2655.0000
##  [8,] 3803.0000 3799.0000 3795.0000
##  [9,] 1550.0000 1491.0000 1432.0000
## [10,] 7266.8947 7206.7895 7146.6842
## [11,] 4622.7771 4565.5542 4508.3313
## [12,] 6093.0000 6093.0000 6093.0000
## [13,] 3371.7368 3333.4737 3295.2105
## [14,] 4112.0000 4075.0000 4038.0000
## [15,] 8939.6842 8881.3684 8823.0526
## [16,] 9488.0000 9488.0000 9488.0000
## [17,] 5632.5051 5618.3979 5604.0400
## [18,] 6561.4320 6561.0631 6560.8234
## [19,] 8539.2159 8408.3856 8263.0920
## [20,] 4771.7336 4784.4024 4792.1433
```

```
## [21,]  781.0846  773.1692  765.2538
## [22,] 2869.0000 2889.0000 2909.0000
## [23,] 8296.2820 8272.5931 8255.4417
## [24,] 3730.1059 3716.7951 3711.7246
## [25,] 4195.0000 4149.0000 4103.0000
## [26,] 3635.0000 3626.0000 3617.0000
## [27,] 2509.5316 2505.5683 2503.7003
## [28,]  511.9809  506.9619  501.9428
## [29,]  545.4211  536.8421  528.2632
## [30,] 1815.7409 1780.1002 1743.5535
```

```
##Import and vidualize the complete population data##################
pop0 <- as_tibble(readxl::read_excel("/Users/yuehenghu/Desktop/RP/RP/population.xlsx",range = "Sheet1!B
pop0
```

```
## # A tibble: 30 x 23
##    `1997` `1998` `1999` `2000` `2001` `2002` `2003` `2004` `2005` `2006` `2007`
##     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1  1309.  1326.  1344.   1364   1385   1423   1456   1493   1538   1601   1676
##  2   994.   996.   998.   1001   1004   1007   1011   1024   1043   1075   1115
##  3  6552.  6593.  6633.   6674   6699   6735   6769   6809   6851   6898   6943
##  4  3208.  3221.  3234.   3247   3272   3294   3314   3335   3355   3375   3393
##  5  2345   2354   2363    2372   2381   2384   2386   2393   2403   2415   2429
##  6  4154   4164   4174    4184   4194   4203   4210   4217   4221   4271   4298
##  7  2655   2664   2673    2682   2691   2699   2704   2709   2716   2723   2730
##  8  3795   3799   3803    3807   3811   3813   3815   3817   3820   3823   3824
##  9  1432   1491   1550    1609   1668   1713   1766   1835   1890   1964   2064
## 10  7147.  7207.  7267.   7327   7359   7406   7458   7523   7588   7656   7723
## # ... with 20 more rows, and 12 more variables: `2008` <dbl>, `2009` <dbl>,
## #   `2010` <dbl>, `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>,
## #   `2015` <dbl>, `2016` <dbl>, `2017` <dbl>, `2018` <dbl>, `2019` <dbl>
```

```
pop0 <- t(pop0)
pop0
```

```
##          [,1]      [,2]     [,3]     [,4] [,5] [,6] [,7] [,8] [,9]    [,10]
## 1997 1308.803  994.3688 6551.947 3207.526 2345 4154 2655 3795 1432 7146.684
## 1998 1325.987  996.2430 6592.632 3220.684 2354 4164 2664 3799 1491 7206.789
## 1999 1344.358  998.4354 6633.316 3233.842 2363 4174 2673 3803 1550 7266.895
## 2000 1364.000 1001.0000 6674.000 3247.000 2372 4184 2682 3807 1609 7327.000
## 2001 1385.000 1004.0000 6699.000 3272.000 2381 4194 2691 3811 1668 7359.000
## 2002 1423.000 1007.0000 6735.000 3294.000 2384 4203 2699 3813 1713 7406.000
## 2003 1456.000 1011.0000 6769.000 3314.000 2386 4210 2704 3815 1766 7458.000
## 2004 1493.000 1024.0000 6809.000 3335.000 2393 4217 2709 3817 1835 7523.000
## 2005 1538.000 1043.0000 6851.000 3355.000 2403 4221 2716 3820 1890 7588.000
## 2006 1601.000 1075.0000 6898.000 3375.000 2415 4271 2723 3823 1964 7656.000
## 2007 1676.000 1115.0000 6943.000 3393.000 2429 4298 2730 3824 2064 7723.000
## 2008 1771.000 1176.0000 6989.000 3411.000 2444 4315 2734 3825 2141 7762.000
## 2009 1860.000 1228.0000 7034.000 3427.000 2458 4341 2740 3826 2210 7810.000
## 2010 1962.000 1299.0000 7194.000 3574.000 2472 4375 2747 3833 2303 7869.000
## 2011 2024.000 1341.0000 7232.000 3562.000 2470 4379 2725 3782 2356 8023.000
## 2012 2078.000 1378.0000 7262.000 3548.000 2464 4375 2698 3724 2399 8120.000
## 2013 2125.000 1410.0000 7288.000 3535.000 2455 4365 2668 3666 2448 8192.000
```

```
## 2014 2171.000 1429.0000 7323.000 3528.000 2449 4358 2642 3608 2467 8281.000
## 2015 2188.000 1439.0000 7345.000 3519.000 2440 4338 2613 3529 2458 8315.000
## 2016 2195.000 1443.0000 7375.000 3514.000 2436 4327 2567 3463 2467 8381.000
## 2017 2194.000 1410.0000 7409.000 3510.000 2433 4312 2526 3399 2466 8423.000
## 2018 2192.000 1383.0000 7426.000 3502.000 2422 4291 2484 3327 2475 8446.000
## 2019 2190.000 1385.0000 7447.000 3497.000 2415 4277 2448 3255 2481 8469.000
##         [,11] [,12]    [,13] [,14]     [,15] [,16]    [,17]    [,18]     [,19]
## 1997 4508.331  6093 3295.211  4038  8823.053  9488 5604.040 6560.823  8263.092
## 1998 4565.554  6093 3333.474  4075  8881.368  9488 5618.398 6561.063  8408.386
## 1999 4622.777  6093 3371.737  4112  8939.684  9488 5632.505 6561.432  8539.216
## 2000 4680.000  6093 3410.000  4149  8998.000  9488 5646.000 6562.000  8650.000
## 2001 4729.000  6128 3445.000  4186  9041.000  9555 5658.000 6596.000  8733.000
## 2002 4776.000  6144 3476.000  4222  9082.000  9613 5672.000 6629.000  8842.000
## 2003 4857.000  6163 3502.000  4254  9125.000  9667 5685.000 6663.000  8963.000
## 2004 4925.000  6228 3529.000  4284  9180.000  9717 5698.000 6698.000  9111.000
## 2005 4991.000  6120 3557.000  4311  9248.000  9380 5710.000 6326.000  9194.000
## 2006 5072.000  6110 3585.000  4339  9309.000  9392 5693.000 6342.000  9442.000
## 2007 5155.000  6118 3612.000  4368  9367.000  9360 5699.000 6355.000  9660.000
## 2008 5212.000  6135 3639.000  4400  9417.000  9429 5711.000 6380.000  9893.000
## 2009 5276.000  6131 3666.000  4432  9470.000  9487 5720.000 6406.000 10130.000
## 2010 5447.000  5957 3693.000  4462  9588.000  9405 5728.000 6570.000 10441.000
## 2011 5570.000  5972 3784.000  4474  9665.000  9461 5760.000 6581.000 10756.000
## 2012 5685.000  5978 3841.000  4475  9708.000  9532 5781.000 6590.000 11041.000
## 2013 5784.000  5988 3885.000  4476  9746.000  9573 5798.000 6600.000 11270.000
## 2014 5890.000  5997 3945.000  4480  9808.000  9645 5816.000 6611.000 11489.000
## 2015 5985.000  6011 3984.000  4485  9866.000  9701 5850.000 6615.000 11678.000
## 2016 6072.000  6033 4016.000  4496  9973.000  9778 5885.000 6625.000 11908.000
## 2017 6170.000  6057 4065.000  4511 10033.000  9829 5904.000 6633.000 12141.000
## 2018 6273.000  6076 4104.000  4513 10077.000  9864 5917.000 6635.000 12348.000
## 2019 6375.000  6092 4137.000  4516 10106.000  9901 5927.000 6640.000 12489.000
##         [,20]    [,21] [,22]    [,23]    [,24] [,25] [,26]    [,27]    [,28]
## 1997 4792.143 765.2538  2909 8255.442 3711.725  4103  3617 2503.700 501.9428
## 1998 4784.402 773.1692  2889 8272.593 3716.795  4149  3626 2505.568 506.9619
## 1999 4771.734 781.0846  2869 8296.282 3730.106  4195  3635 2509.532 511.9809
## 2000 4751.000 789.0000  2849 8329.000 3756.000  4241  3644 2515.000 517.0000
## 2001 4788.000 796.0000  2829 8143.000 3799.000  4287  3653 2523.000 523.0000
## 2002 4822.000 803.0000  2814 8110.000 3837.000  4333  3662 2531.000 529.0000
## 2003 4857.000 811.0000  2803 8176.000 3870.000  4376  3672 2537.000 534.0000
## 2004 4889.000 818.0000  2793 8090.000 3904.000  4415  3681 2541.000 539.0000
## 2005 4660.000 828.0000  2798 8212.000 3730.000  4450  3690 2545.000 543.0000
## 2006 4719.000 836.0000  2808 8169.000 3690.000  4483  3699 2547.000 548.0000
## 2007 4768.000 845.0000  2816 8127.000 3632.000  4514  3708 2548.000 552.0000
## 2008 4816.000 854.0000  2839 8138.000 3596.000  4543  3718 2551.000 554.0000
## 2009 4856.000 864.0000  2859 8185.000 3537.000  4571  3727 2555.000 557.0000
## 2010 4610.000 869.0000  2885 8045.000 3479.000  4602  3735 2560.000 563.0000
## 2011 4655.000 890.0000  2944 8064.000 3530.000  4620  3765 2552.000 568.0000
## 2012 4694.000 910.0000  2975 8085.000 3587.000  4631  3787 2550.000 571.0000
## 2013 4731.000 920.0000  3011 8109.000 3632.000  4641  3804 2537.000 571.0000
## 2014 4770.000 936.0000  3043 8139.000 3677.000  4653  3827 2531.000 576.0000
## 2015 4811.000 945.0000  3070 8196.000 3708.000  4663  3846 2523.000 577.0000
## 2016 4857.000 957.0000  3110 8251.000 3758.000  4677  3874 2520.000 582.0000
## 2017 4907.000 972.0000  3144 8289.000 3803.000  4693  3904 2522.000 586.0000
## 2018 4947.000 982.0000  3163 8321.000 3822.000  4703  3931 2515.000 587.0000
## 2019 4982.000 995.0000  3188 8351.000 3848.000  4714  3944 2509.000 590.0000
```

```
##          [,29]    [,30]
## 1997 528.2632 1743.553
## 1998 536.8421 1780.100
## 1999 545.4211 1815.741
## 2000 554.0000 1849.000
## 2001 563.0000 1876.000
## 2002 572.0000 1905.000
## 2003 580.0000 1934.000
## 2004 588.0000 1963.000
## 2005 596.0000 2010.000
## 2006 604.0000 2050.000
## 2007 610.0000 2095.000
## 2008 618.0000 2131.000
## 2009 625.0000 2159.000
## 2010 633.0000 2185.000
## 2011 648.0000 2225.000
## 2012 659.0000 2253.000
## 2013 666.0000 2285.000
## 2014 678.0000 2325.000
## 2015 684.0000 2385.000
## 2016 695.0000 2428.000
## 2017 705.0000 2480.000
## 2018 710.0000 2520.000
## 2019 717.0000 2559.000
```

```r
province <- population$Region[1:30]
province
```

```
##  [1] "Beijing"        "Tianjin"        "Hebei"          "Shanxi"
##  [5] "Inner Mongolia" "Liaoning"       "Jilin"          "Heilongjiang"
##  [9] "Shanghai"       "Jiangsu"        "Zhejiang"       "Anhui"
## [13] "Fujian"         "Jiangxi"        "Shandong"       "Henan"
## [17] "Hubei"          "Hunan"          "Guangdong"      "Guangxi"
## [21] "Hainan"         "Chongqing"      "Sichuan"        "Guizhou"
## [25] "Yunnan"         "Shaanxi"        "Gansu"          "Qinghai"
## [29] "Ningxia"        "Xinjiang"
```

```r
province -> colnames(pop0)
province
```

```
##  [1] "Beijing"        "Tianjin"        "Hebei"          "Shanxi"
##  [5] "Inner Mongolia" "Liaoning"       "Jilin"          "Heilongjiang"
##  [9] "Shanghai"       "Jiangsu"        "Zhejiang"       "Anhui"
## [13] "Fujian"         "Jiangxi"        "Shandong"       "Henan"
## [17] "Hubei"          "Hunan"          "Guangdong"      "Guangxi"
## [21] "Hainan"         "Chongqing"      "Sichuan"        "Guizhou"
## [25] "Yunnan"         "Shaanxi"        "Gansu"          "Qinghai"
## [29] "Ningxia"        "Xinjiang"
```

```r
as_tibble(pop0) %>%
  mutate(Year =rownames(pop0)) %>%
  select(Year, c(province))-> pop0_df
pop0_df
```

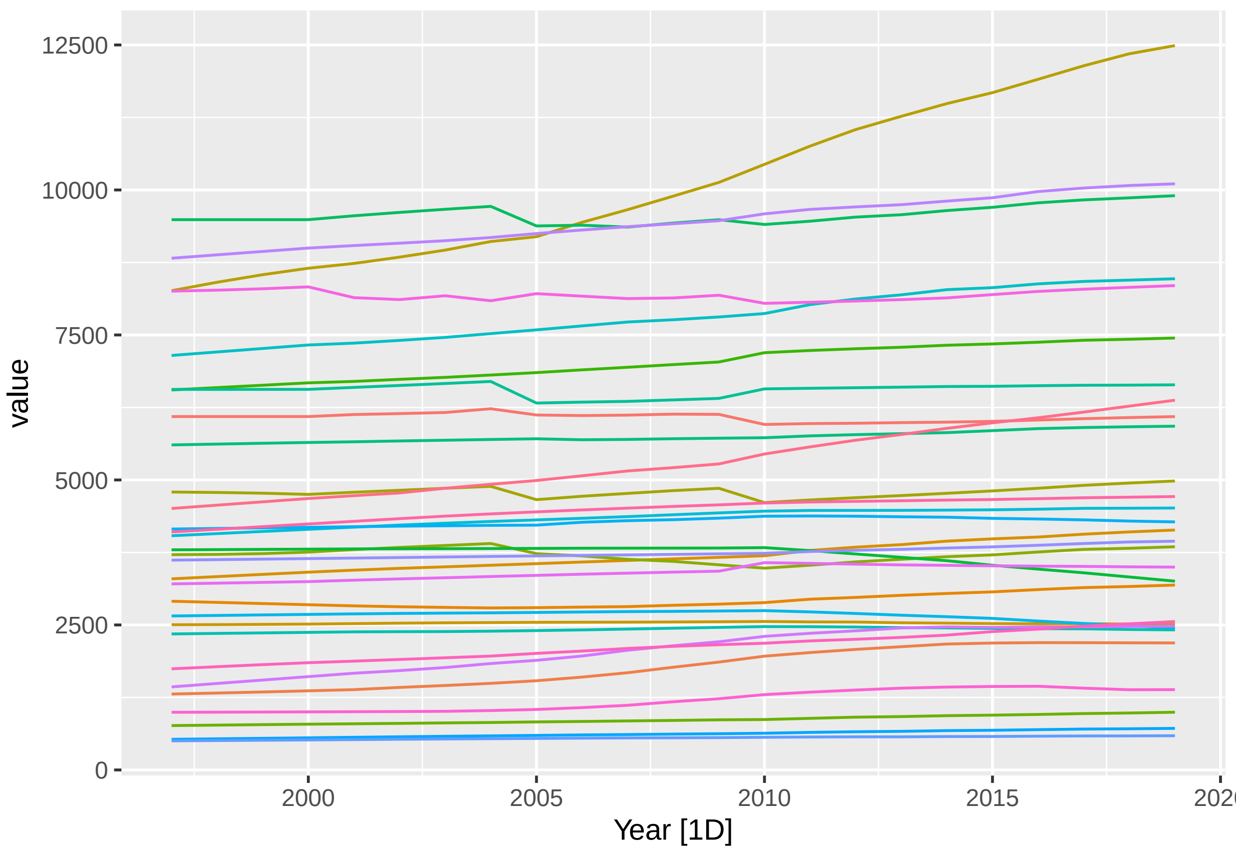
```
## # A tibble: 23 x 31
##    Year  Beijing Tianjin Hebei Shanxi Inner Mong~1 Liaon~2 Jilin Heilo~3 Shang~4
##    <chr>   <dbl>   <dbl> <dbl>  <dbl>        <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1 1997    1309.    994. 6552.  3208.         2345    4154  2655    3795    1432
##  2 1998    1326.    996. 6593.  3221.         2354    4164  2664    3799    1491
##  3 1999    1344.    998. 6633.  3234.         2363    4174  2673    3803    1550
##  4 2000    1364    1001  6674   3247          2372    4184  2682    3807    1609
##  5 2001    1385    1004  6699   3272          2381    4194  2691    3811    1668
##  6 2002    1423    1007  6735   3294          2384    4203  2699    3813    1713
##  7 2003    1456    1011  6769   3314          2386    4210  2704    3815    1766
##  8 2004    1493    1024  6809   3335          2393    4217  2709    3817    1835
##  9 2005    1538    1043  6851   3355          2403    4221  2716    3820    1890
## 10 2006    1601    1075  6898   3375          2415    4271  2723    3823    1964
## # ... with 13 more rows, 21 more variables: Jiangsu <dbl>, Zhejiang <dbl>,
## #   Anhui <dbl>, Fujian <dbl>, Jiangxi <dbl>, Shandong <dbl>, Henan <dbl>,
## #   Hubei <dbl>, Hunan <dbl>, Guangdong <dbl>, Guangxi <dbl>, Hainan <dbl>,
## #   Chongqing <dbl>, Sichuan <dbl>, Guizhou <dbl>, Yunnan <dbl>, Shaanxi <dbl>,
## #   Gansu <dbl>, Qinghai <dbl>, Ningxia <dbl>, Xinjiang <dbl>, and abbreviated
## #   variable names 1: `Inner Mongolia`, 2: Liaoning, 3: Heilongjiang,
## #   4: Shanghai
```

```
Year =rownames(pop0)
as_tibble(pop0)%>%
  mutate(Year = paste0(Year, "-01-01")) %>%
  mutate(Year = ymd(Year)) %>%
  as_tsibble(index = Year)  -> pop0_ts
pop0_ts
```

```
## # A tsibble: 23 x 31 [1D]
##    Beijing Tianjin Hebei Shanxi Inner Mo~1 Liaon~2 Jilin Heilo~3 Shang~4 Jiangsu
##      <dbl>   <dbl> <dbl>  <dbl>      <dbl>   <dbl> <dbl>   <dbl>   <dbl>   <dbl>
##  1   1309.    994. 6552.  3208.       2345    4154  2655    3795    1432   7147.
##  2   1326.    996. 6593.  3221.       2354    4164  2664    3799    1491   7207.
##  3   1344.    998. 6633.  3234.       2363    4174  2673    3803    1550   7267.
##  4   1364    1001  6674   3247        2372    4184  2682    3807    1609   7327
##  5   1385    1004  6699   3272        2381    4194  2691    3811    1668   7359
##  6   1423    1007  6735   3294        2384    4203  2699    3813    1713   7406
##  7   1456    1011  6769   3314        2386    4210  2704    3815    1766   7458
##  8   1493    1024  6809   3335        2393    4217  2709    3817    1835   7523
##  9   1538    1043  6851   3355        2403    4221  2716    3820    1890   7588
## 10   1601    1075  6898   3375        2415    4271  2723    3823    1964   7656
## # ... with 13 more rows, 21 more variables: Zhejiang <dbl>, Anhui <dbl>,
## #   Fujian <dbl>, Jiangxi <dbl>, Shandong <dbl>, Henan <dbl>, Hubei <dbl>,
## #   Hunan <dbl>, Guangdong <dbl>, Guangxi <dbl>, Hainan <dbl>, Chongqing <dbl>,
## #   Sichuan <dbl>, Guizhou <dbl>, Yunnan <dbl>, Shaanxi <dbl>, Gansu <dbl>,
## #   Qinghai <dbl>, Ningxia <dbl>, Xinjiang <dbl>, Year <date>, and abbreviated
## #   variable names 1: `Inner Mongolia`, 2: Liaoning, 3: Heilongjiang,
## #   4: Shanghai
```

```
pop0_ts %>%
  pivot_longer(col = -Year, names_to = "Province") -> pop0_ts_long
autoplot(pop0_ts_long, divideTime=0.5) +
  theme(legend.position = "none")
```

```
## Plot variable not specified, automatically selected `.vars = value`

## Warning: Ignoring unknown parameters: divideTime
```



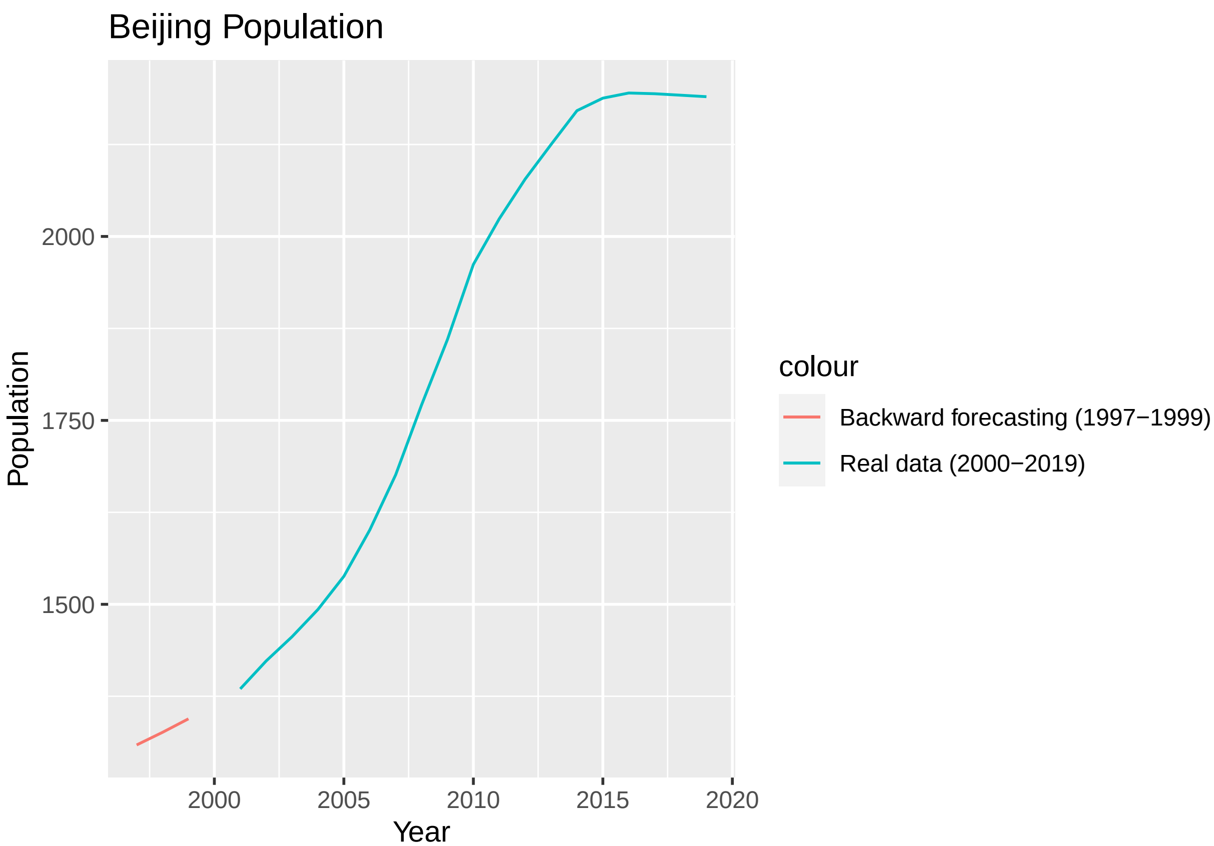

```
pop0_df
```

```
## # A tibble: 23 x 31
##    Year  Beijing Tianjin Hebei Shanxi Inner Mong~1 Liaon~2 Jilin Heilo~3 Shang~4
##    <chr>   <dbl>   <dbl> <dbl>  <dbl>        <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1 1997    1309.    994. 6552.  3208.         2345    4154  2655    3795    1432
##  2 1998    1326.    996. 6593.  3221.         2354    4164  2664    3799    1491
##  3 1999    1344.    998. 6633.  3234.         2363    4174  2673    3803    1550
##  4 2000    1364    1001  6674   3247          2372    4184  2682    3807    1609
##  5 2001    1385    1004  6699   3272          2381    4194  2691    3811    1668
##  6 2002    1423    1007  6735   3294          2384    4203  2699    3813    1713
##  7 2003    1456    1011  6769   3314          2386    4210  2704    3815    1766
##  8 2004    1493    1024  6809   3335          2393    4217  2709    3817    1835
##  9 2005    1538    1043  6851   3355          2403    4221  2716    3820    1890
## 10 2006    1601    1075  6898   3375          2415    4271  2723    3823    1964
## # ... with 13 more rows, 21 more variables: Jiangsu <dbl>, Zhejiang <dbl>,
## #   Anhui <dbl>, Fujian <dbl>, Jiangxi <dbl>, Shandong <dbl>, Henan <dbl>,
## #   Hubei <dbl>, Hunan <dbl>, Guangdong <dbl>, Guangxi <dbl>, Hainan <dbl>,
## #   Chongqing <dbl>, Sichuan <dbl>, Guizhou <dbl>, Yunnan <dbl>, Shaanxi <dbl>,
## #   Gansu <dbl>, Qinghai <dbl>, Ningxia <dbl>, Xinjiang <dbl>, and abbreviated
## #   variable names 1: `Inner Mongolia`, 2: Liaoning, 3: Heilongjiang,
## #   4: Shanghai
```

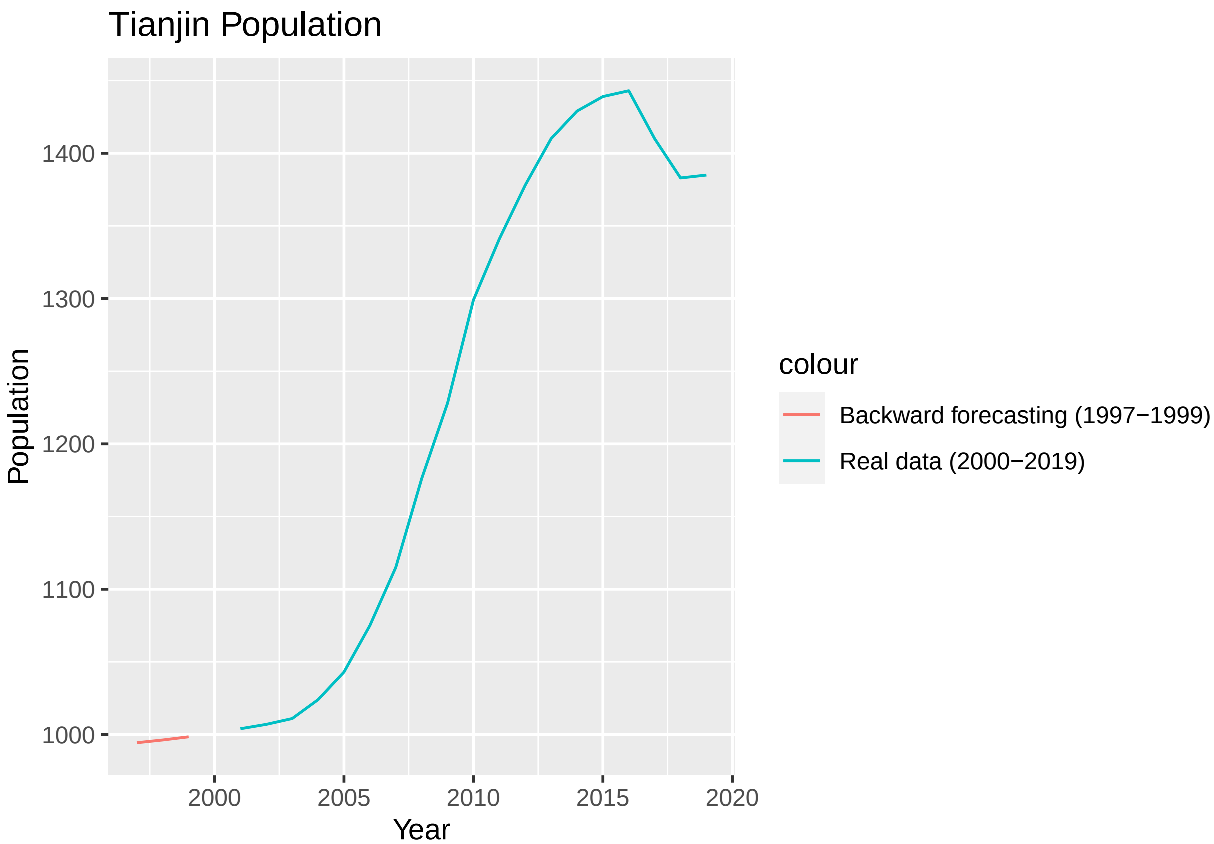
```r
pop0_df %>%
  filter(Year <= "1999") -> before2000
pop0_df %>%
  filter(Year > "2000") -> since2000
pop0_df
```

```
## # A tibble: 23 x 31
##    Year  Beijing Tianjin Hebei Shanxi Inner Mong~1 Liaon~2 Jilin Heilo~3 Shang~4
##    <chr>   <dbl>   <dbl> <dbl>  <dbl>        <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1 1997    1309.    994. 6552.  3208.         2345    4154  2655    3795    1432
##  2 1998    1326.    996. 6593.  3221.         2354    4164  2664    3799    1491
##  3 1999    1344.    998. 6633.  3234.         2363    4174  2673    3803    1550
##  4 2000    1364    1001  6674   3247          2372    4184  2682    3807    1609
##  5 2001    1385    1004  6699   3272          2381    4194  2691    3811    1668
##  6 2002    1423    1007  6735   3294          2384    4203  2699    3813    1713
##  7 2003    1456    1011  6769   3314          2386    4210  2704    3815    1766
##  8 2004    1493    1024  6809   3335          2393    4217  2709    3817    1835
##  9 2005    1538    1043  6851   3355          2403    4221  2716    3820    1890
## 10 2006    1601    1075  6898   3375          2415    4271  2723    3823    1964
## # ... with 13 more rows, 21 more variables: Jiangsu <dbl>, Zhejiang <dbl>,
## #   Anhui <dbl>, Fujian <dbl>, Jiangxi <dbl>, Shandong <dbl>, Henan <dbl>,
## #   Hubei <dbl>, Hunan <dbl>, Guangdong <dbl>, Guangxi <dbl>, Hainan <dbl>,
## #   Chongqing <dbl>, Sichuan <dbl>, Guizhou <dbl>, Yunnan <dbl>, Shaanxi <dbl>,
## #   Gansu <dbl>, Qinghai <dbl>, Ningxia <dbl>, Xinjiang <dbl>, and abbreviated
## #   variable names 1: ‘Inner Mongolia‘, 2: Liaoning, 3: Heilongjiang,
## #   4: Shanghai
```

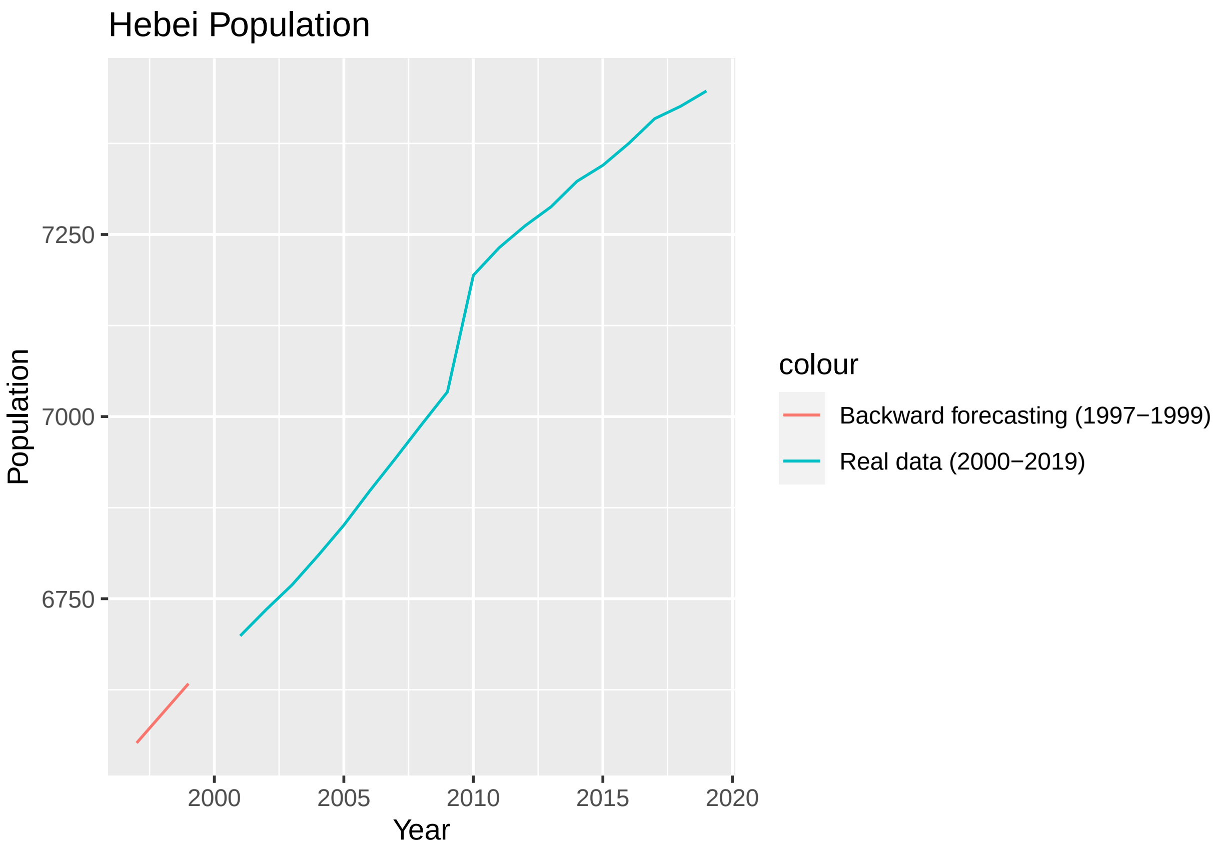
```r
# plot 10 provinces
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Beijing, color = "Backward forecasting (1997-1999)"), data = 
  geom_line(aes(x = as.numeric(Year), y = Beijing, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Beijing Population")
```



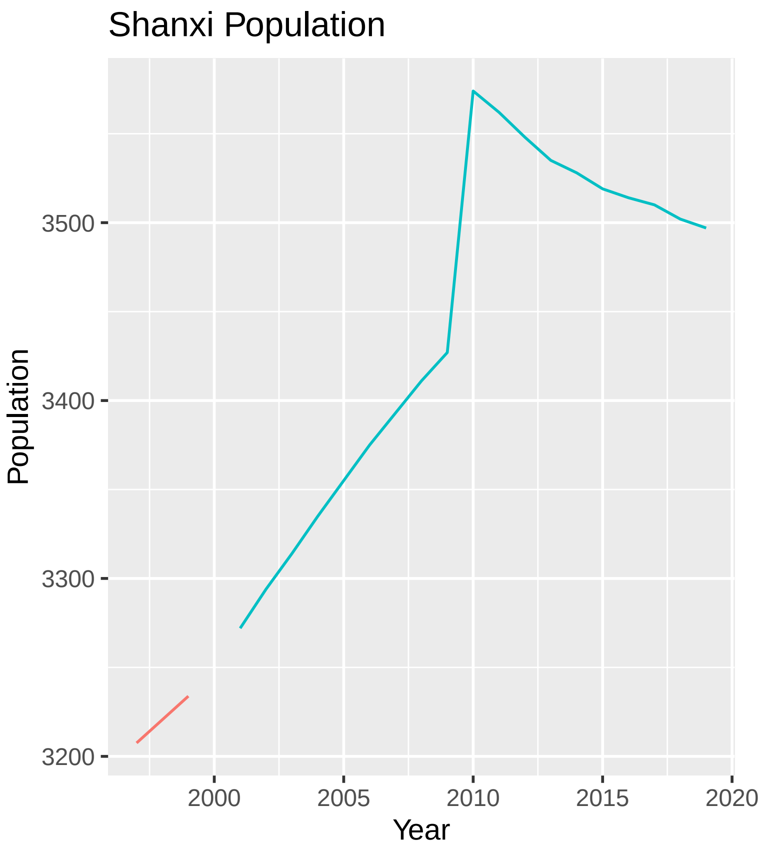

```r
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Tianjin, color = "Backward forecasting (1997-1999)"), data =
  geom_line(aes(x = as.numeric(Year), y = Tianjin, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Tianjin Population")
```



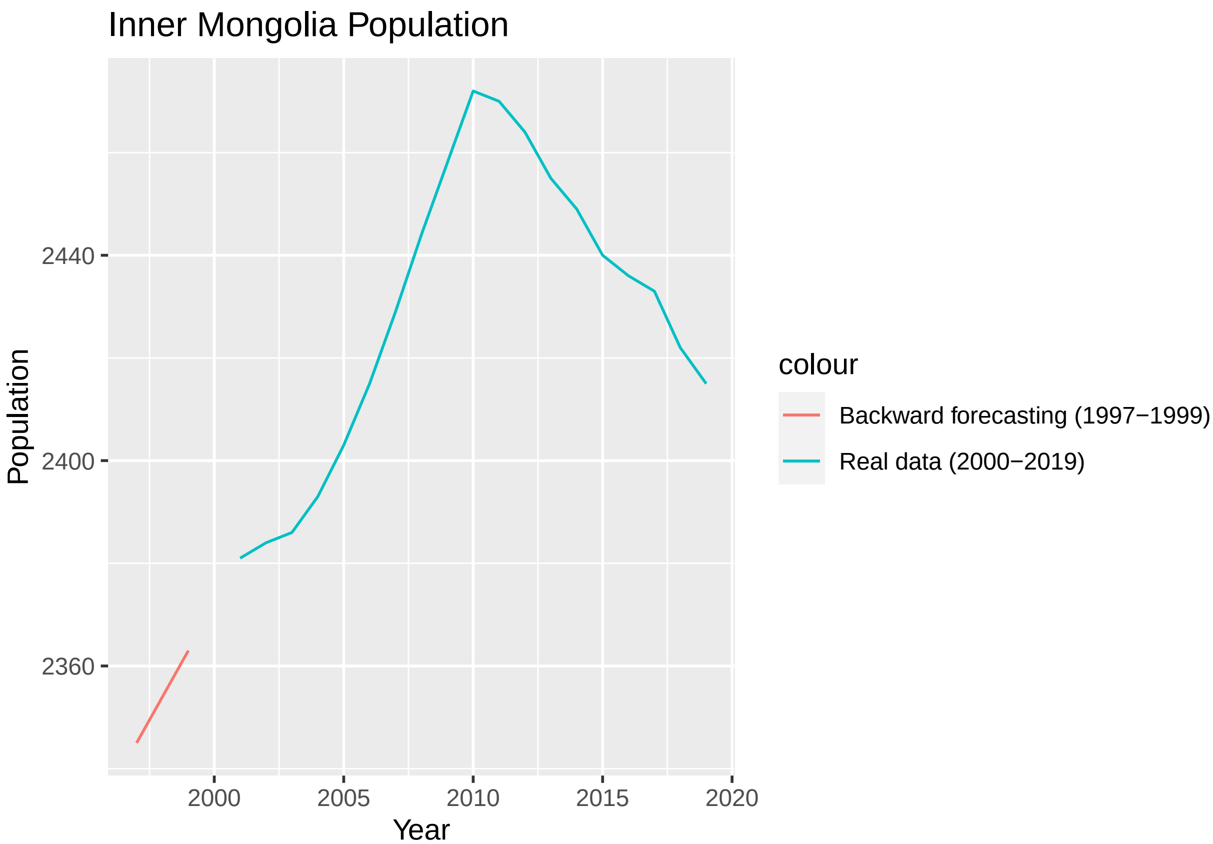

```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Hebei, color = "Backward forecasting (1997-1999)"), data = bei
  geom_line(aes(x = as.numeric(Year), y = Hebei, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Hebei Population")
```



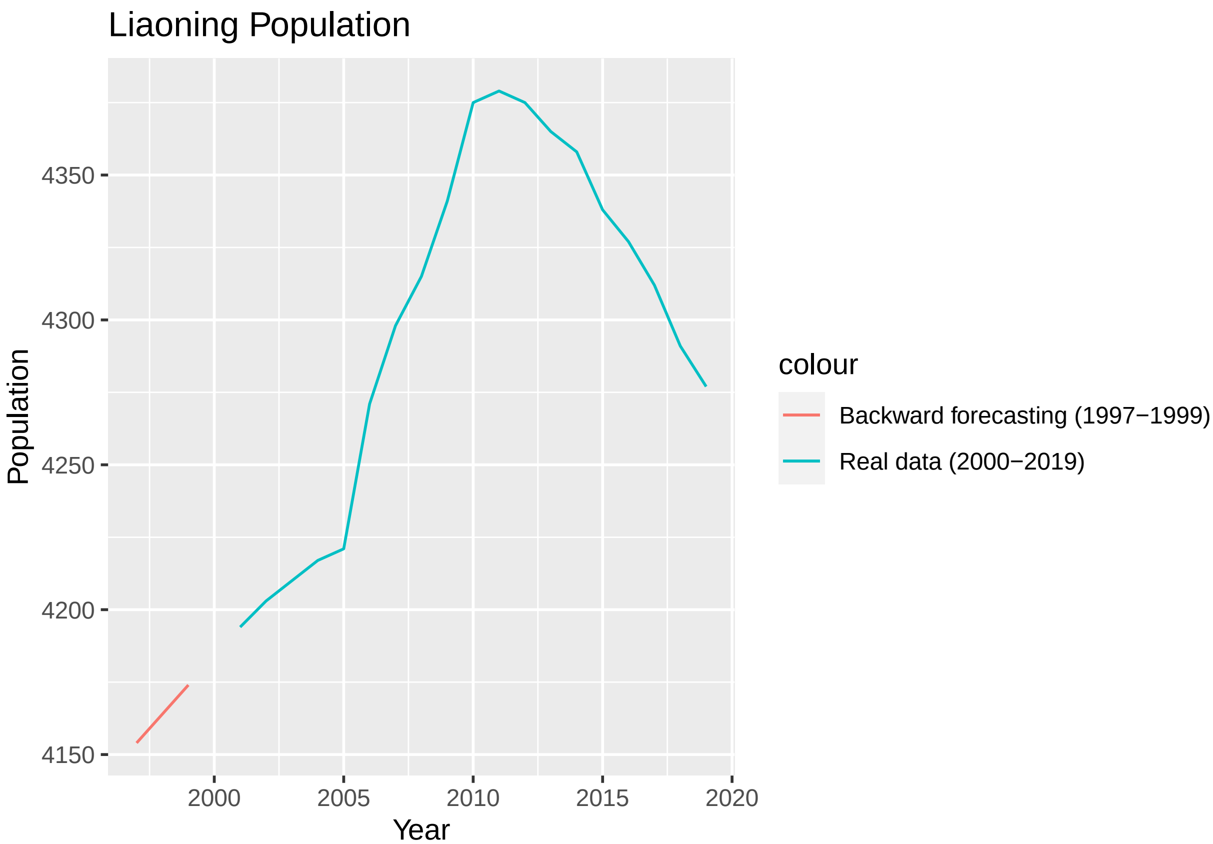

```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Shanxi, color = "Backward forecasting (1997-1999)"), data = be
  geom_line(aes(x = as.numeric(Year), y = Shanxi, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Shanxi Population")
```



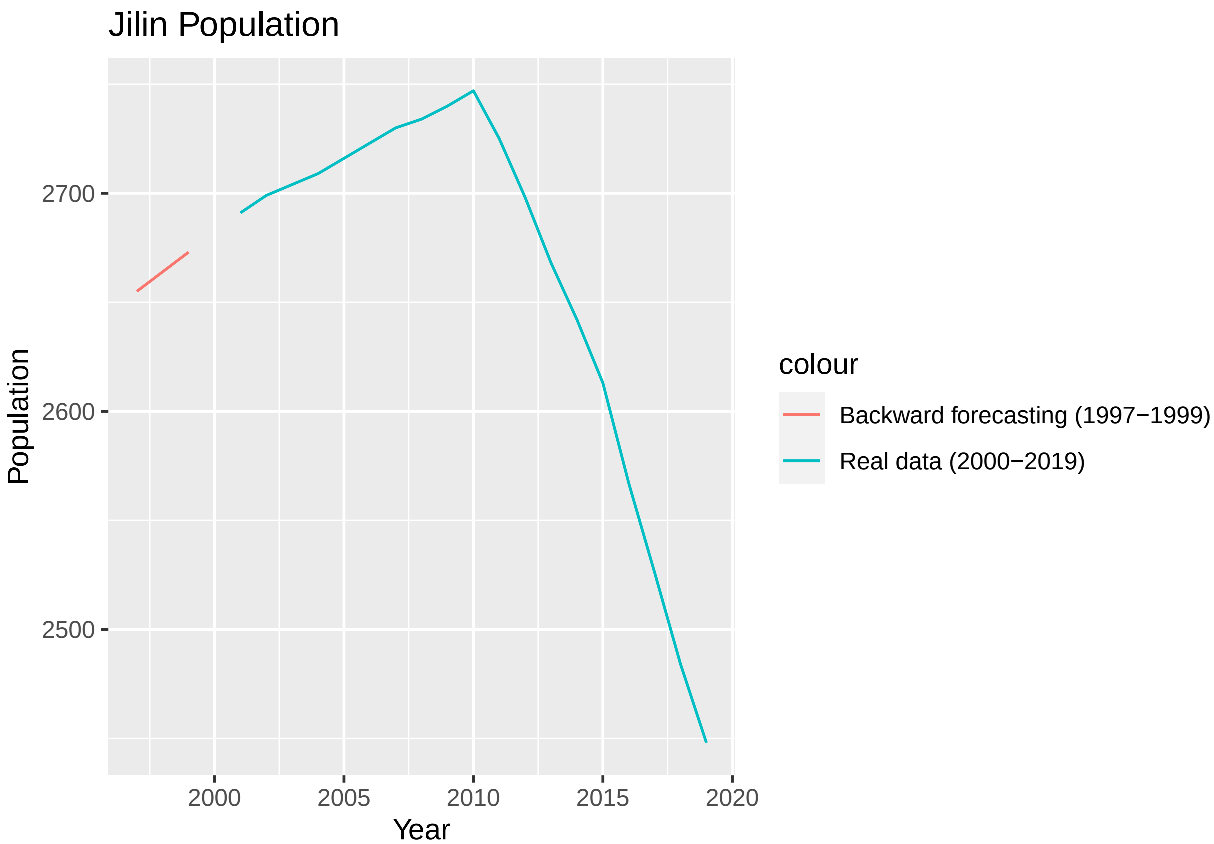

```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = `Inner Mongolia`, color = "Backward forecasting (1997-1999)")
  geom_line(aes(x = as.numeric(Year), y = `Inner Mongolia`, color = "Real data (2000-2019)"), data = si
  xlab("Year")+
  ylab("Population")+
  ggtitle("Inner Mongolia Population")
```



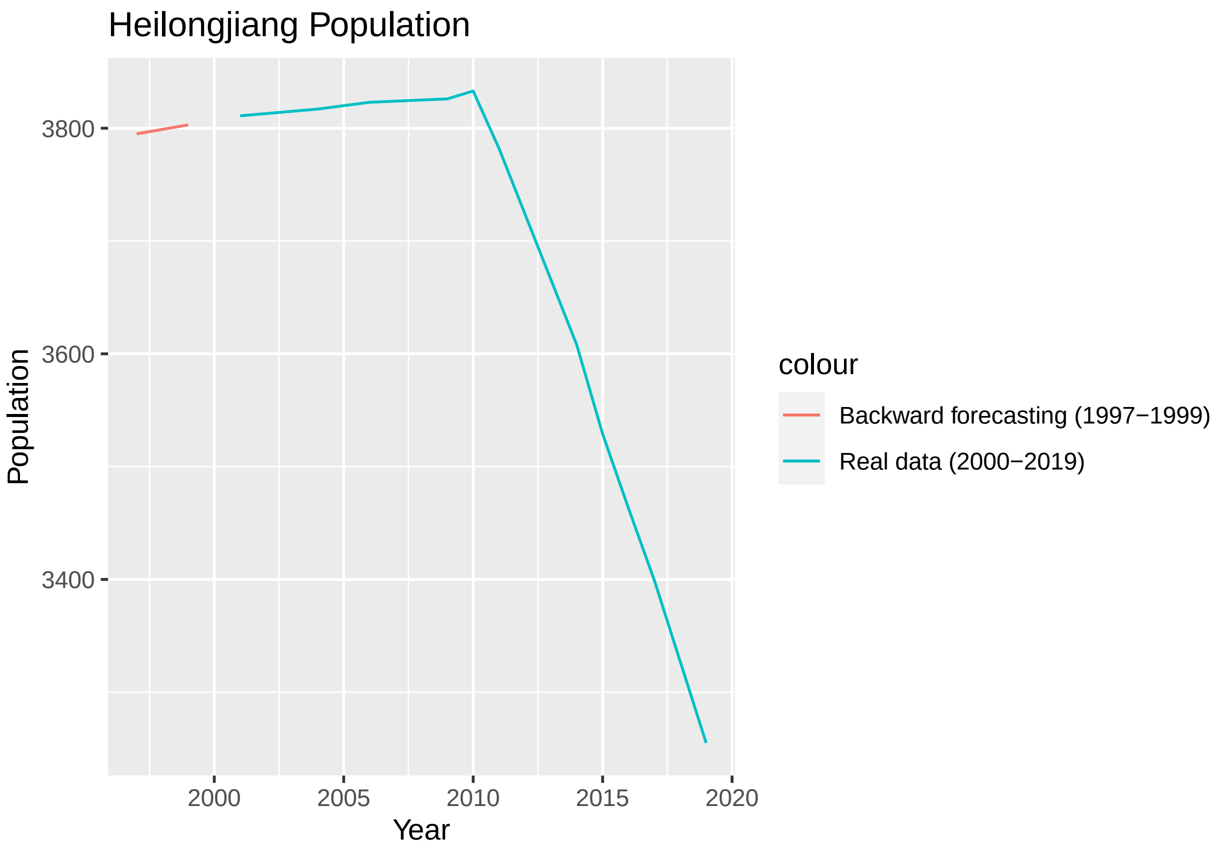

```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Liaoning, color = "Backward forecasting (1997-1999)"), data =
  geom_line(aes(x = as.numeric(Year), y = Liaoning, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Liaoning Population")
```



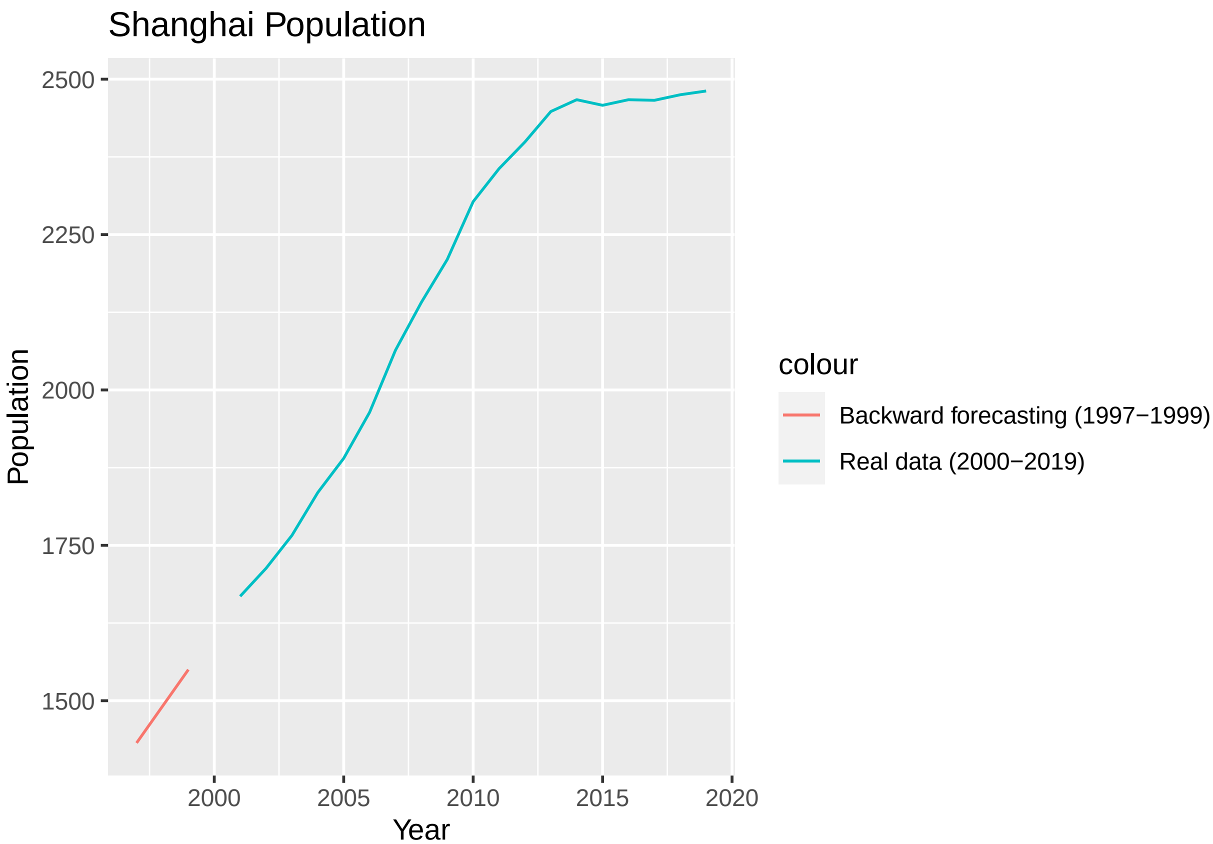

```r
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Jilin, color = "Backward forecasting (1997-1999)"), data = be
  geom_line(aes(x = as.numeric(Year), y = Jilin, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Jilin Population")
```



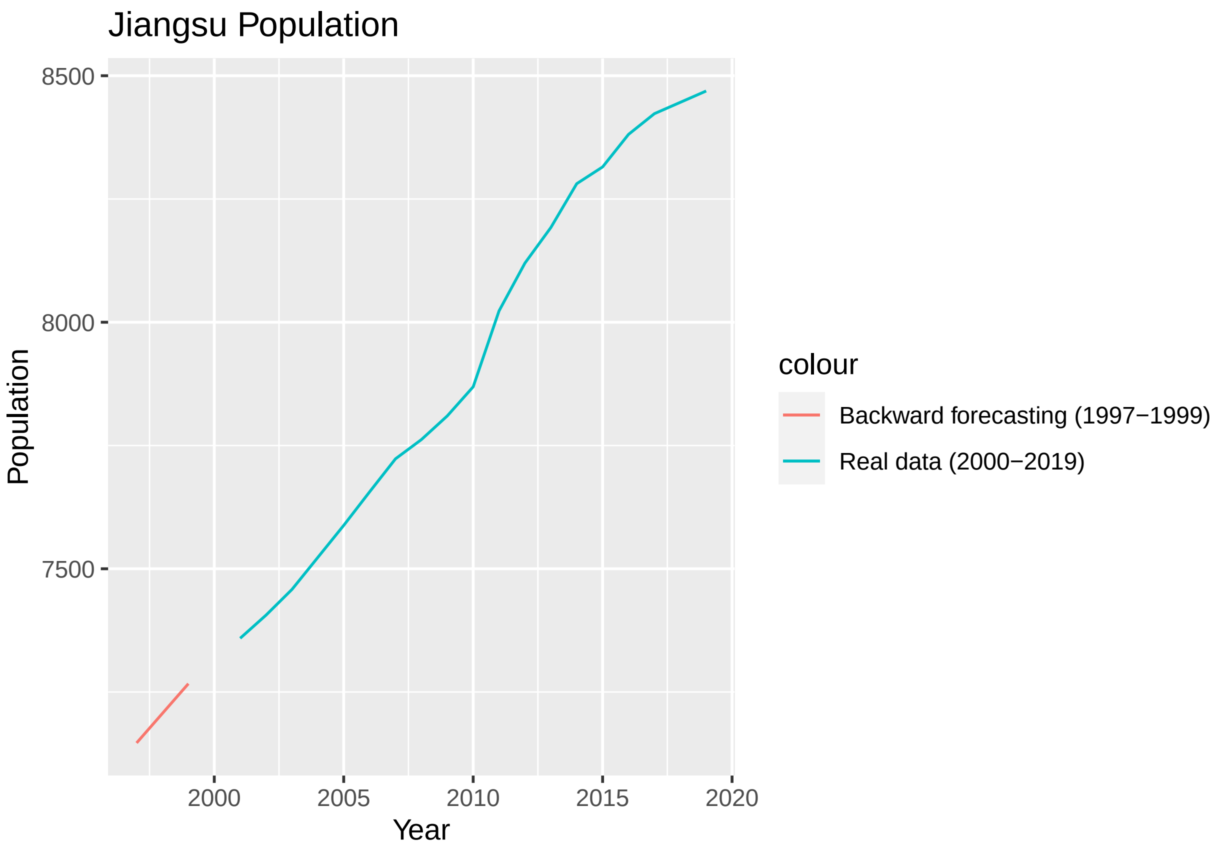

```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Heilongjiang, color = "Backward forecasting (1997-1999)"), dat
  geom_line(aes(x = as.numeric(Year), y = Heilongjiang, color = "Real data (2000-2019)"), data = since2(
  xlab("Year")+
  ylab("Population")+
  ggtitle("Heilongjiang Population")
```



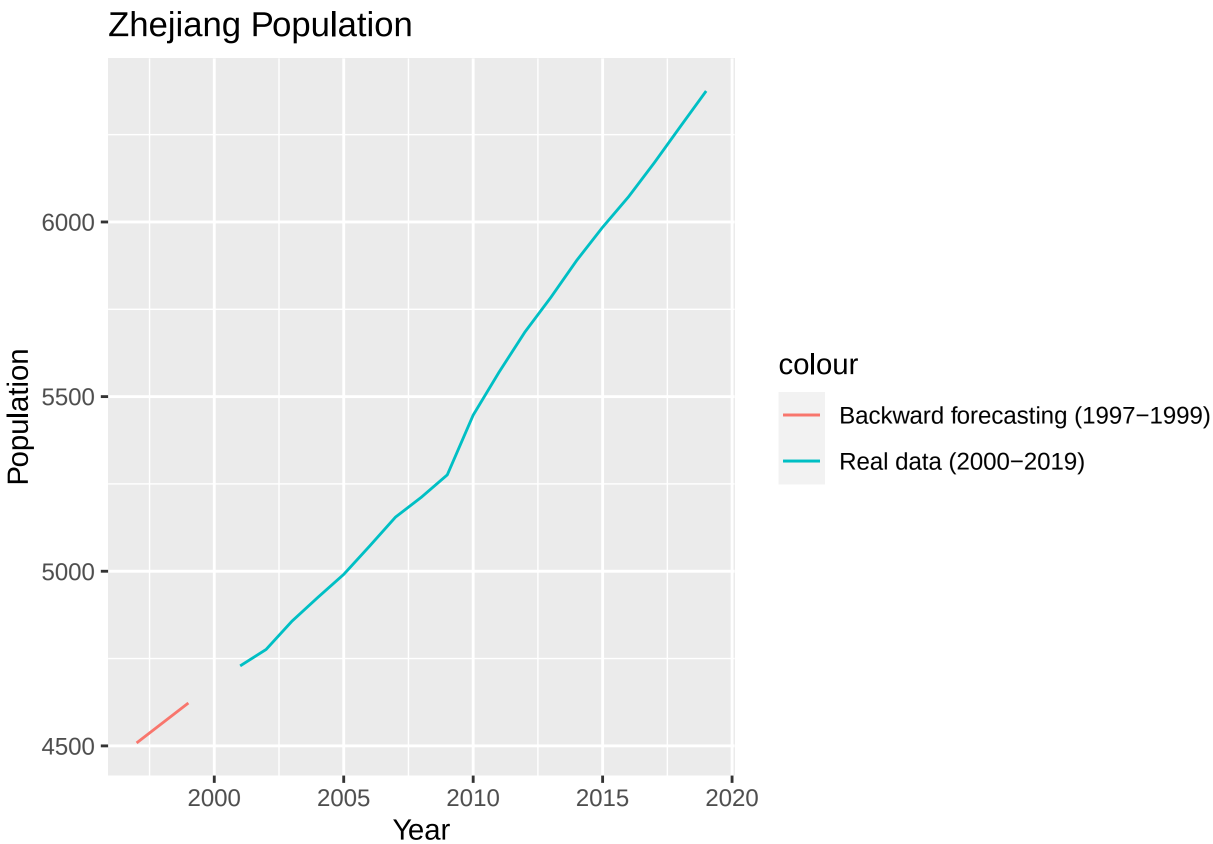

```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Shanghai, color = "Backward forecasting (1997-1999)"), data =
  geom_line(aes(x = as.numeric(Year), y = Shanghai, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Shanghai Population")
```




```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Jiangsu, color = "Backward forecasting (1997-1999)"), data = 
  geom_line(aes(x = as.numeric(Year), y = Jiangsu, color = "Real data (2000-2019)"), data = since2000)+
  xlab("Year")+
  ylab("Population")+
  ggtitle("Jiangsu Population")
```




```
ggplot() +
  geom_line(aes(x = as.numeric(Year), y = Zhejiang, color = "Backward forecasting (1997-1999)"), data =
  geom_line(aes(x = as.numeric(Year), y = Zhejiang, color = "Real data (2000-2019)"), data = since2000)
  xlab("Year")+
  ylab("Population")+
  ggtitle("Zhejiang Population")
```

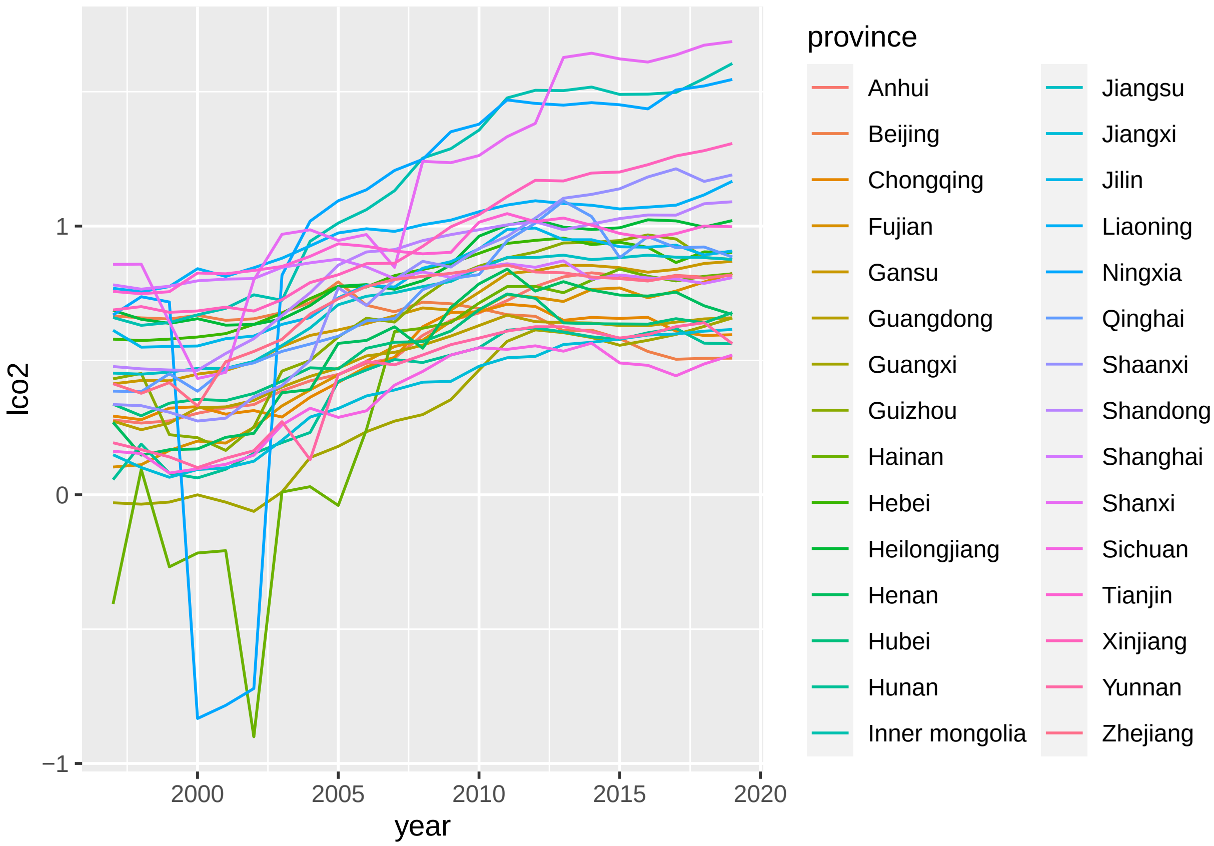


Load the transformed dataset

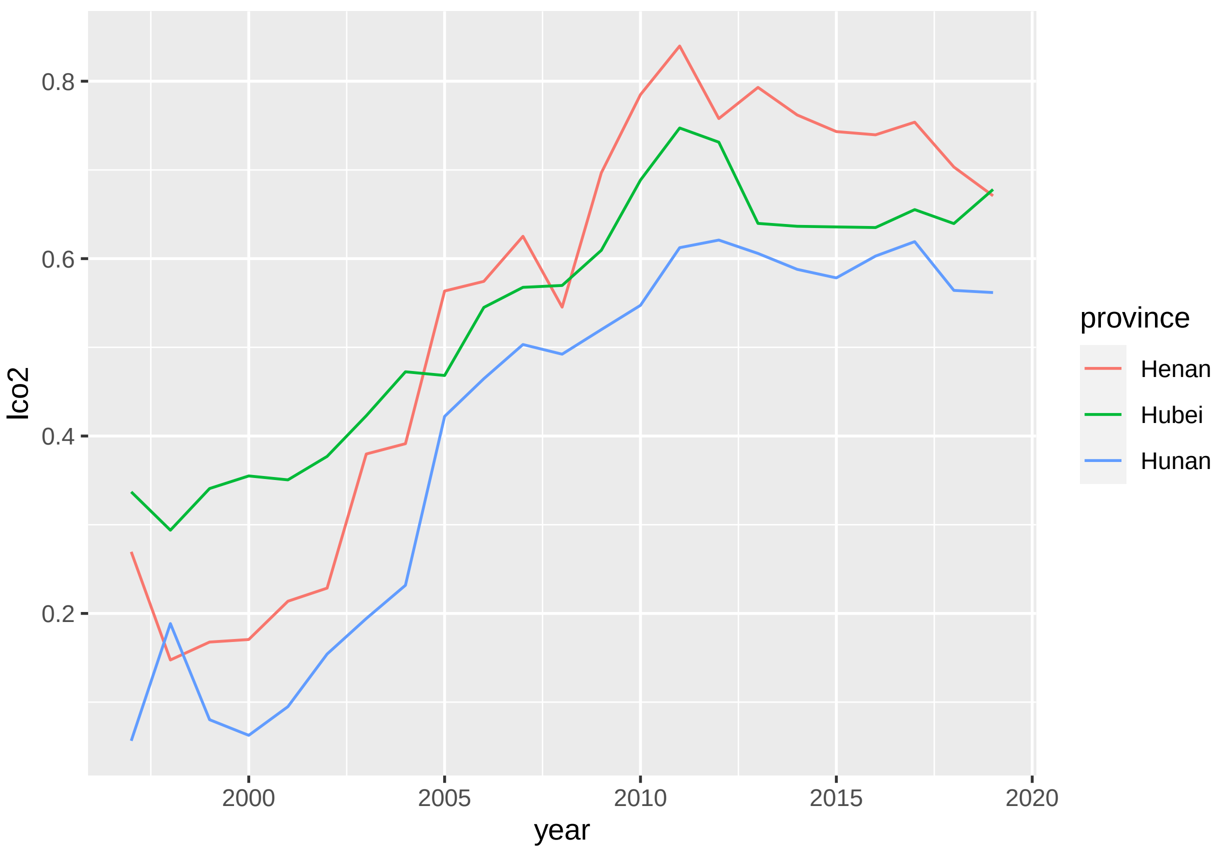
All variables are already in per-capita term, and after log-transformation

Visualizing lco2 over time (log(CO2 per capita))

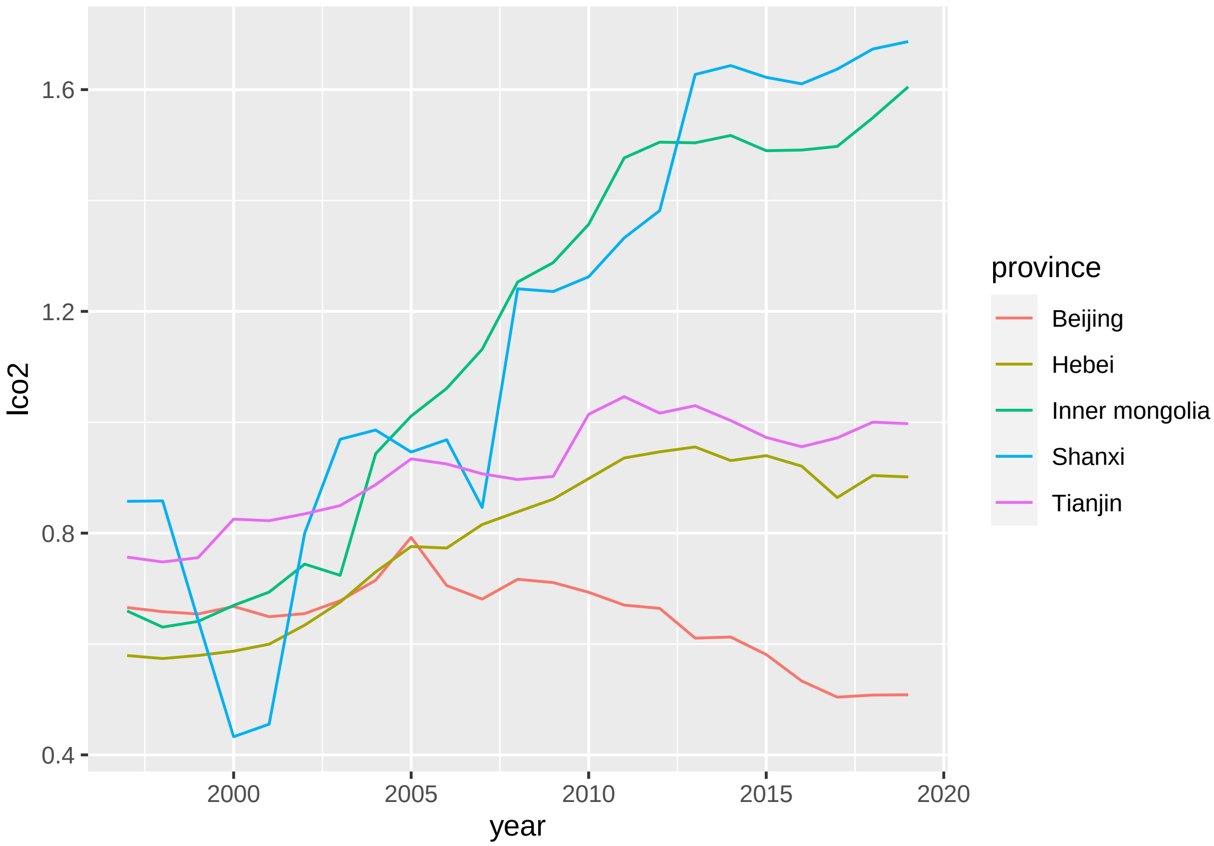
```
all_0915 %>%
  ggplot(aes(x =year, y =lco2, colour = province, gg=TRUE)) +
  geom_line()
```



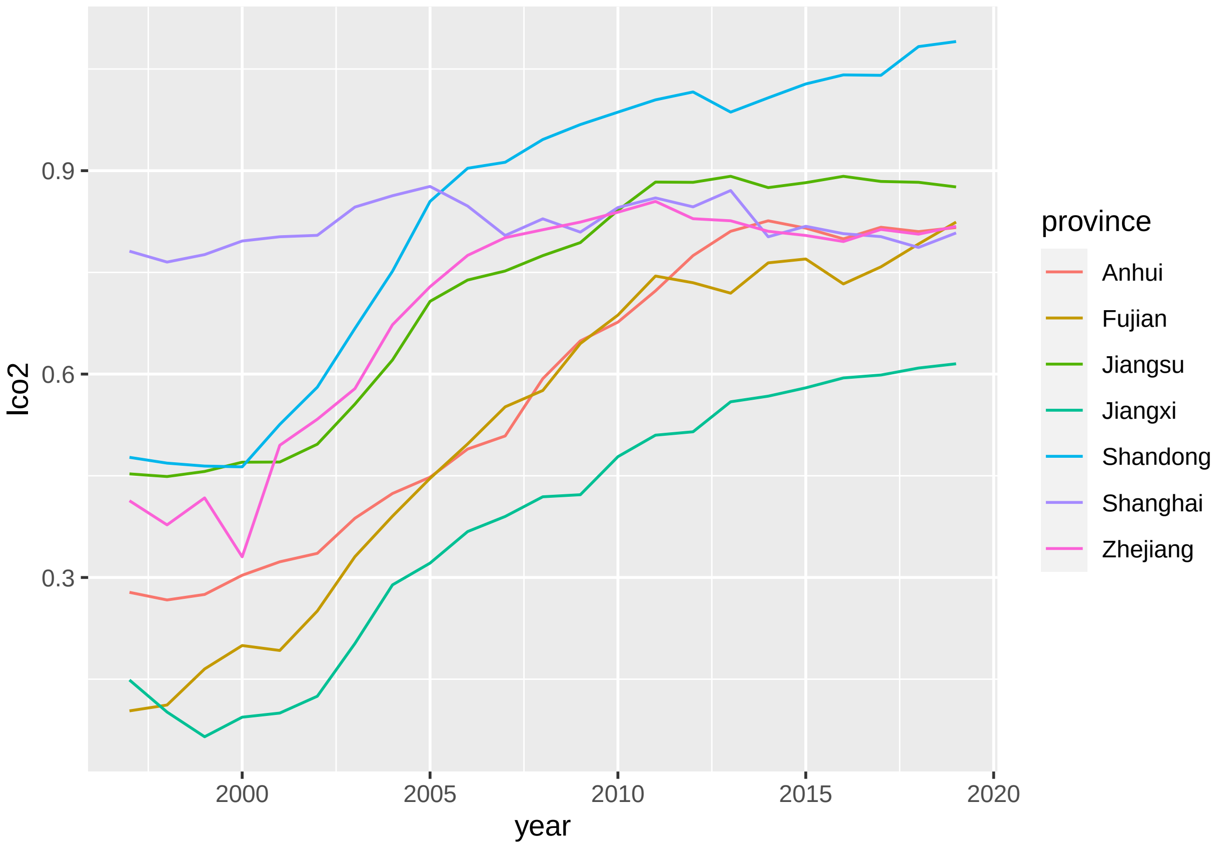

```
all_0915 %>%
  filter(region == "C") %>%
  ggplot(aes(x = year, y = lco2, colour = province)) +
  geom_line()
```



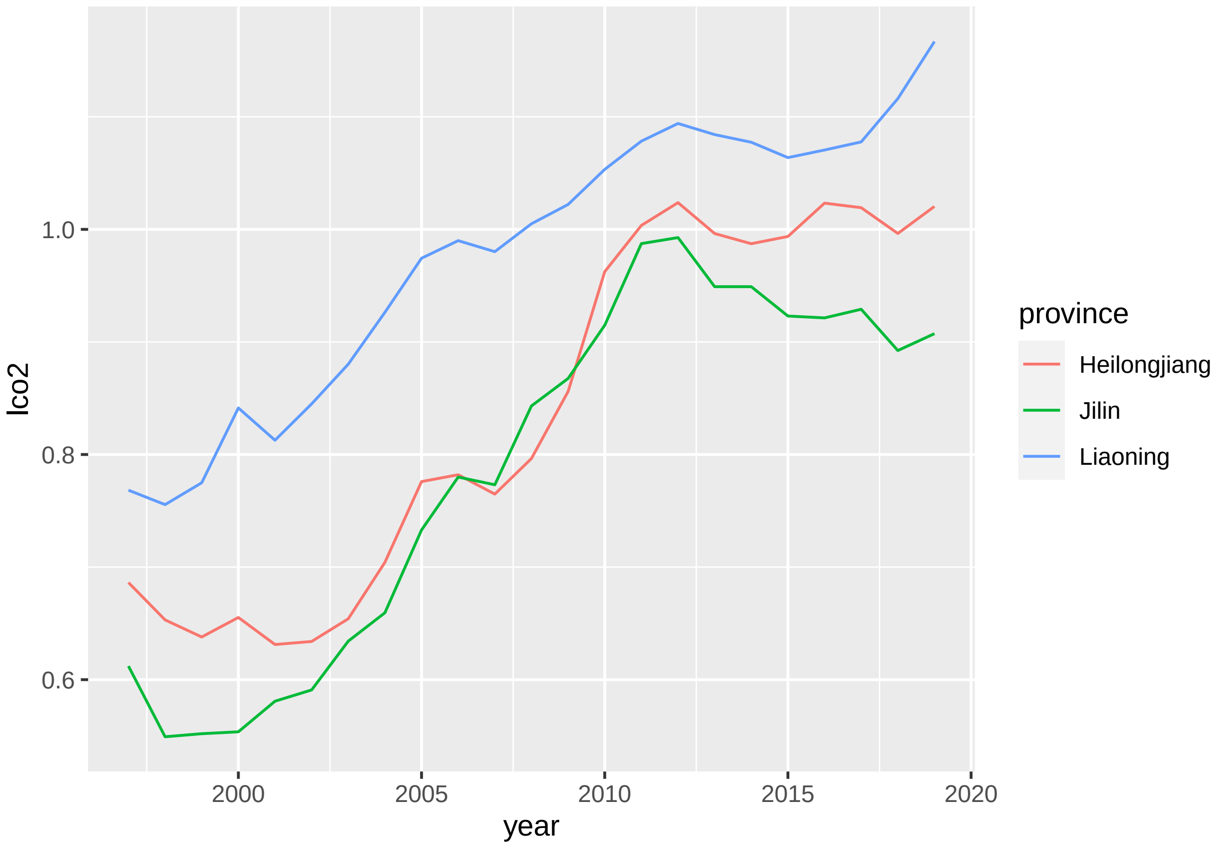

```
all_0915 %>%
  filter(region == "N") %>%
  ggplot(aes(x = year, y = lco2, colour = province)) +
  geom_line()
```



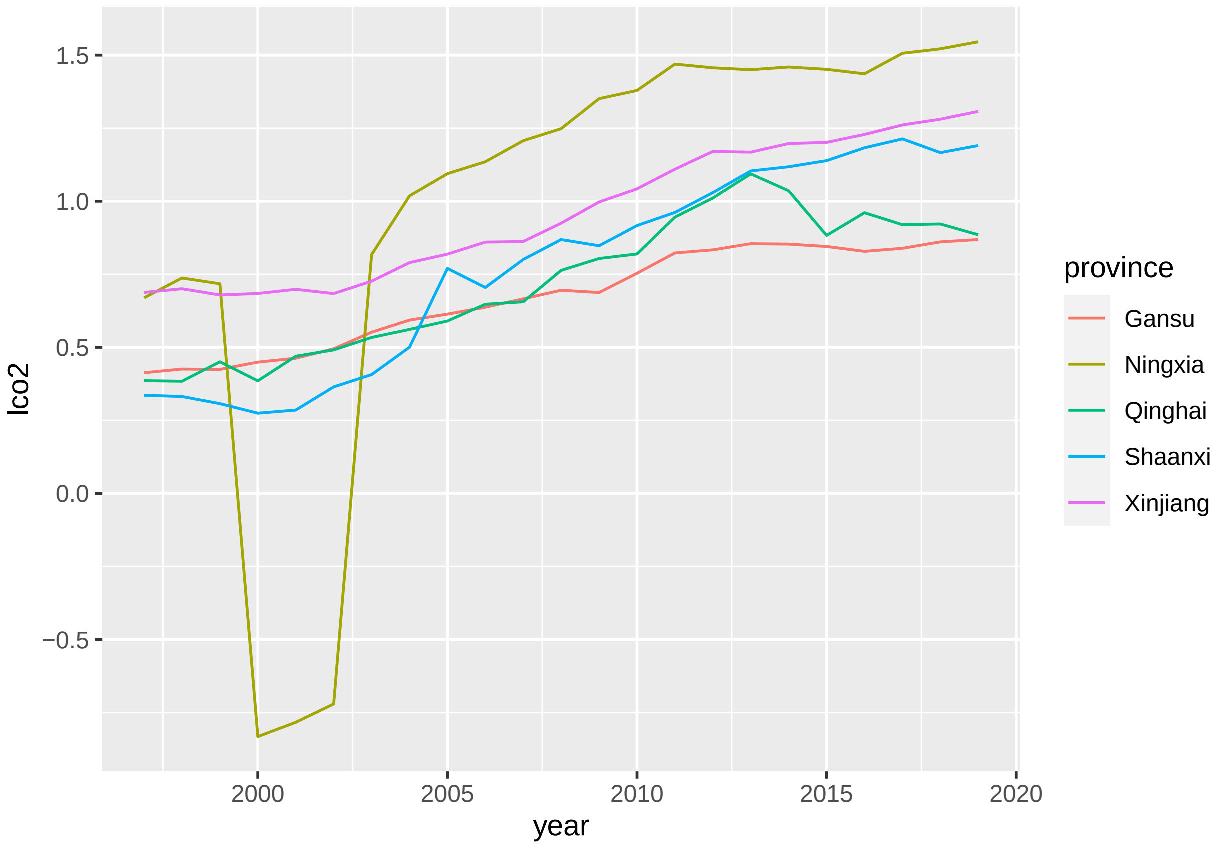

```
all_0915 %>%
  filter(region == "E") %>%
  ggplot(aes(x = year, y = lco2, colour = province)) +
  geom_line()
```



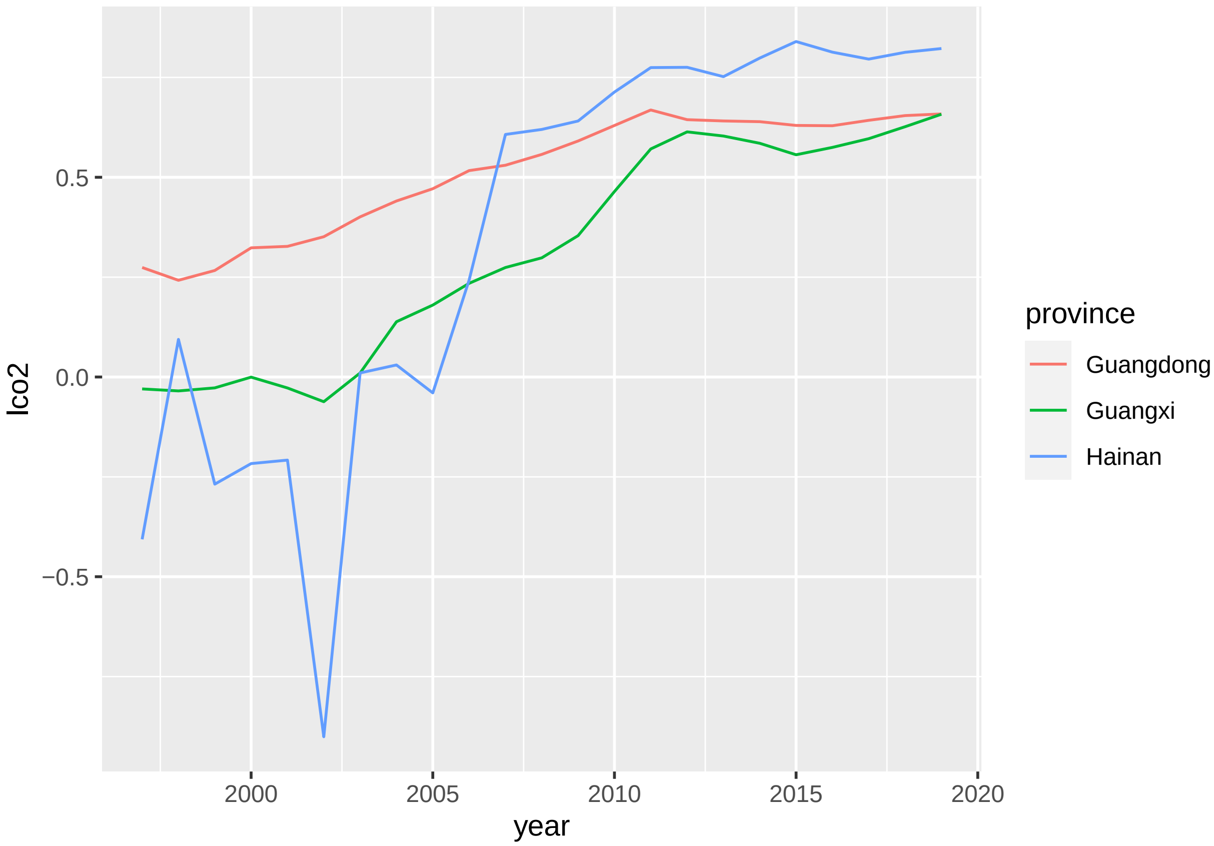

```
all_0915 %>%
  filter(region == "NE") %>%
  ggplot(aes(x = year, y = lco2, colour = province)) +
  geom_line()
```



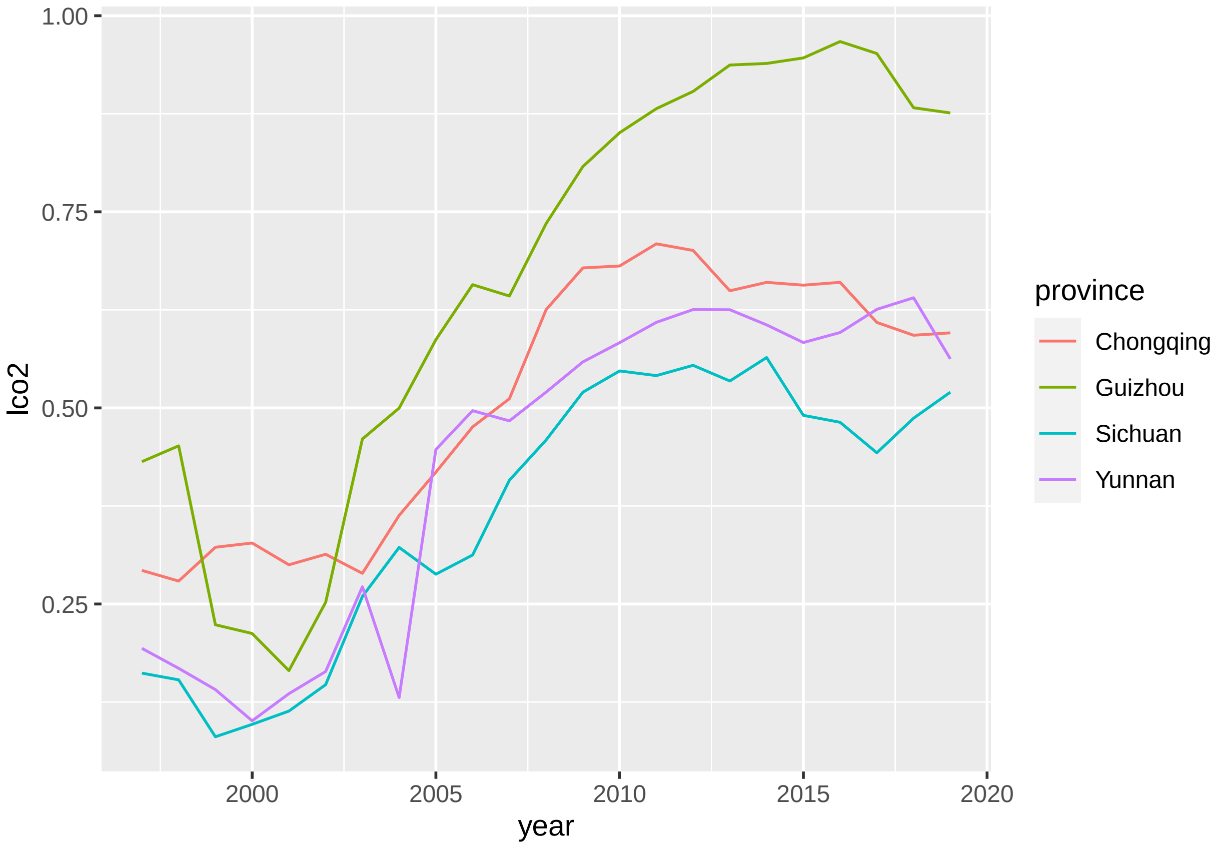

```r
all_0915 %>%
  filter(region == "NW") %>%
  ggplot(aes(x = year, y = lco2, colour = province)) +
  geom_line()
```




```r
all_0915 %>%
  filter(region == "S") %>%
  ggplot(aes(x = year, y = lco2, colour = province)) +
  geom_line()
```

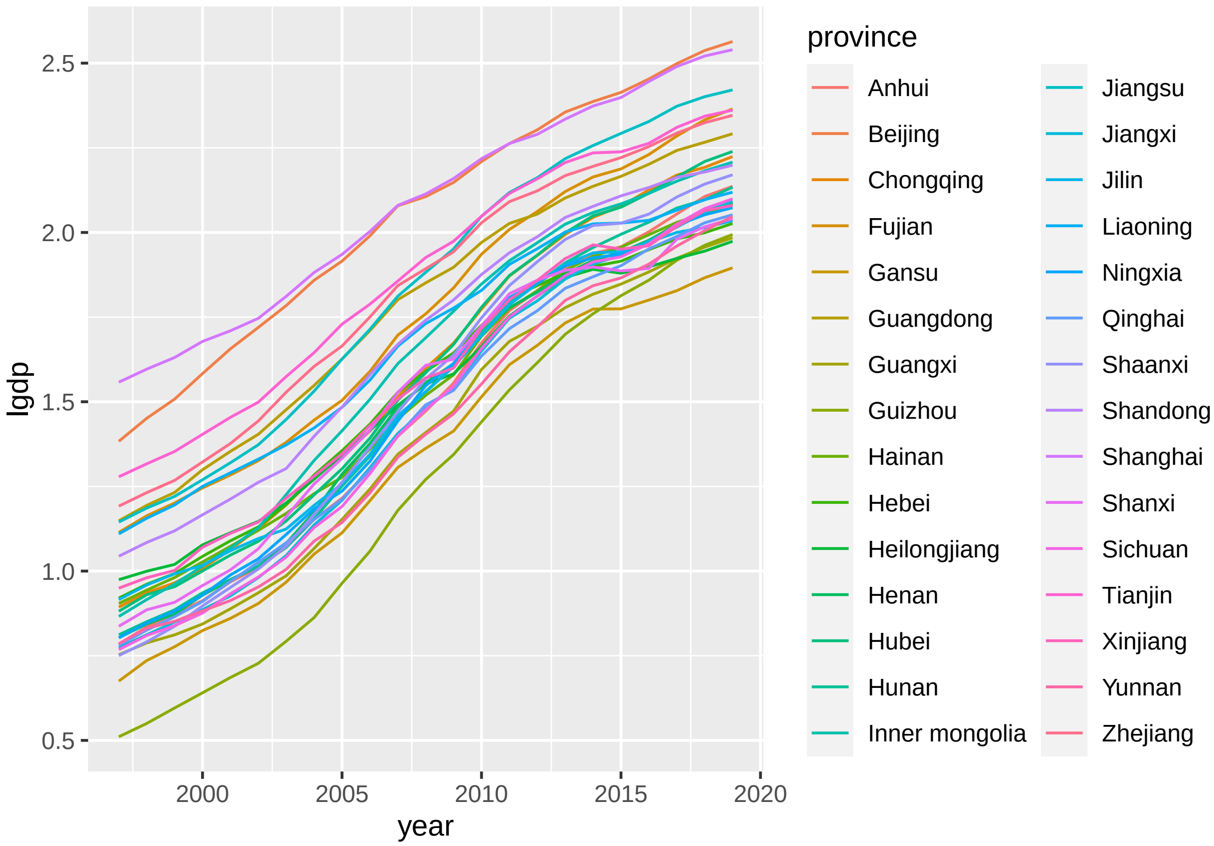



```
all_0915 %>%
  filter(region == "SW") %>%
  ggplot(aes(x = year, y = lco2, colour = province)) +
  geom_line()
```

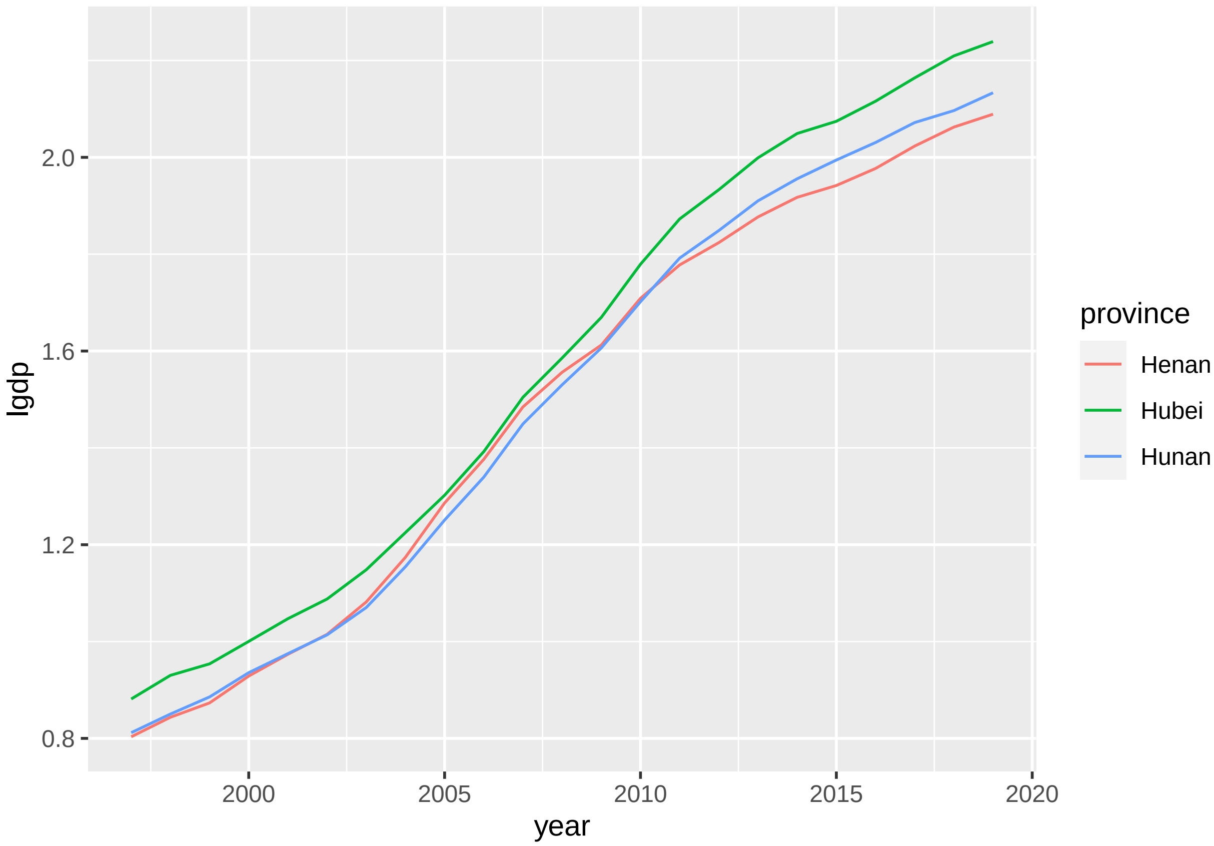


Visualizing lgdp over time (log(GDP per capita))

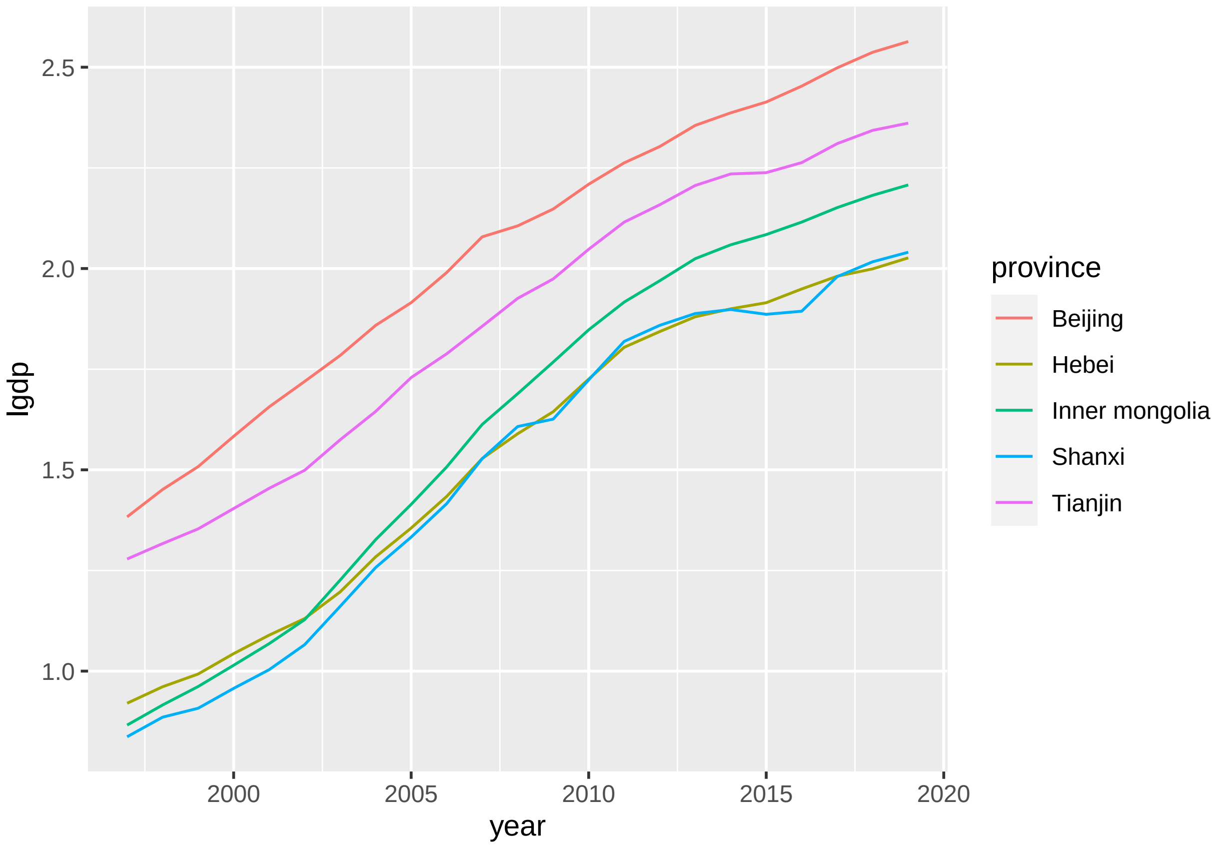
```
all_0915 %>%
  ggplot(aes(x =year, y =lgdp, colour = province, gg=TRUE)) +
  geom_line()
```



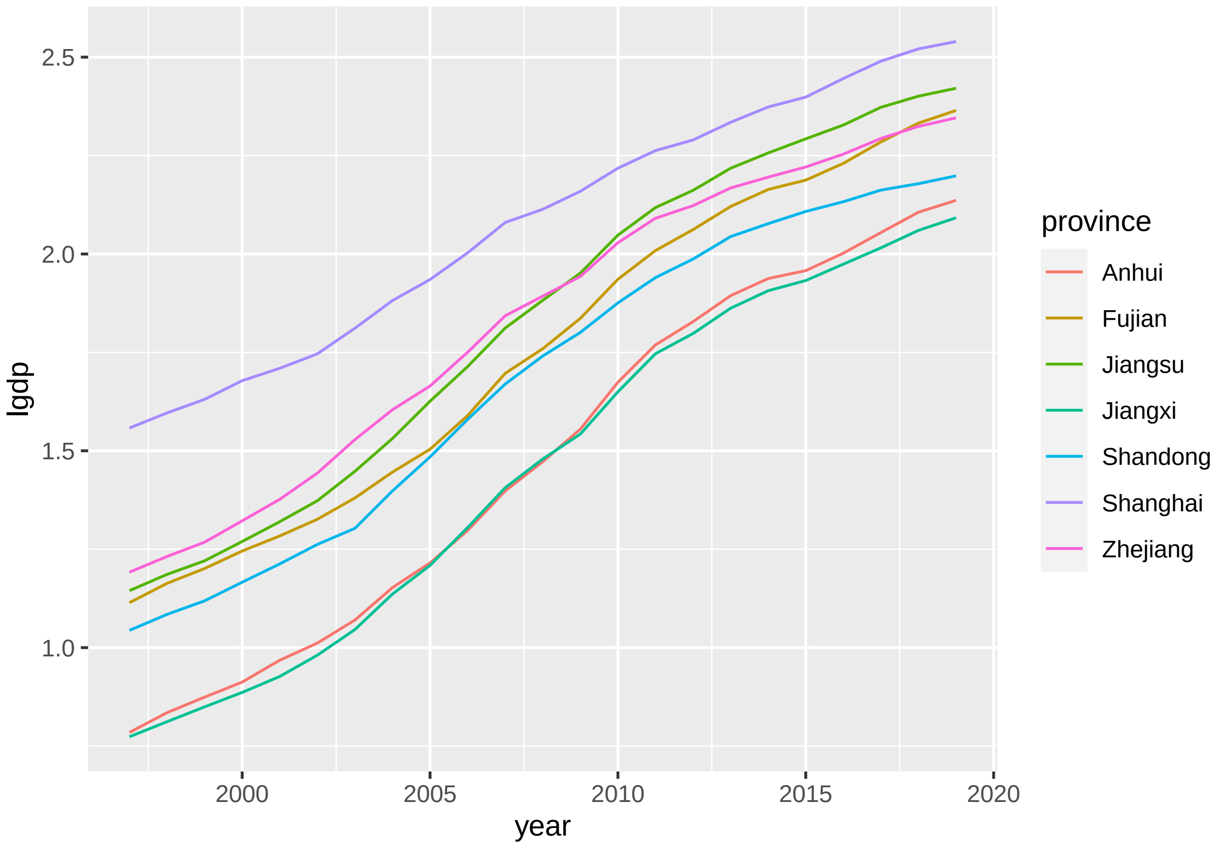

```r
all_0915 %>%
  filter(region == "C") %>%
  ggplot(aes(x = year, y = lgdp, colour = province)) +
  geom_line()
```



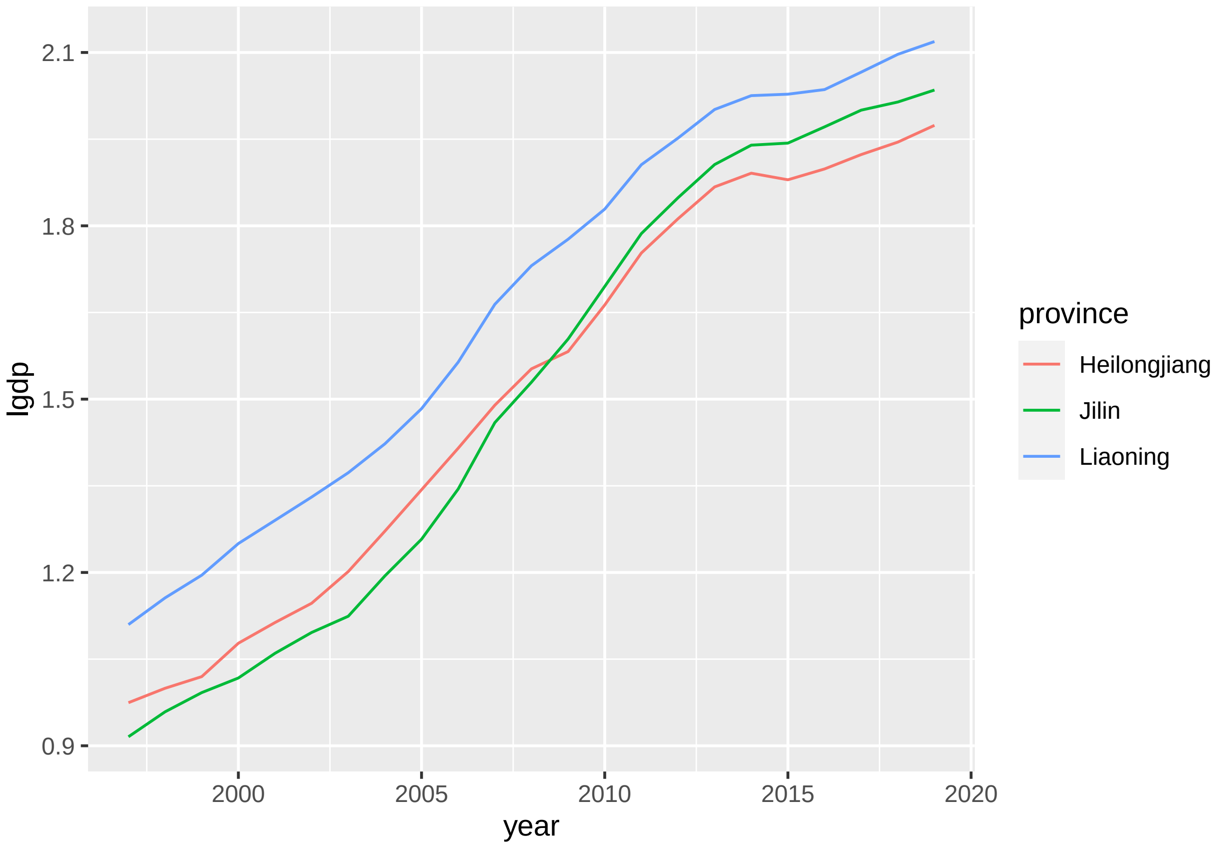

```r
all_0915 %>%
  filter(region == "N") %>%
  ggplot(aes(x = year, y = lgdp, colour = province)) +
  geom_line()
```



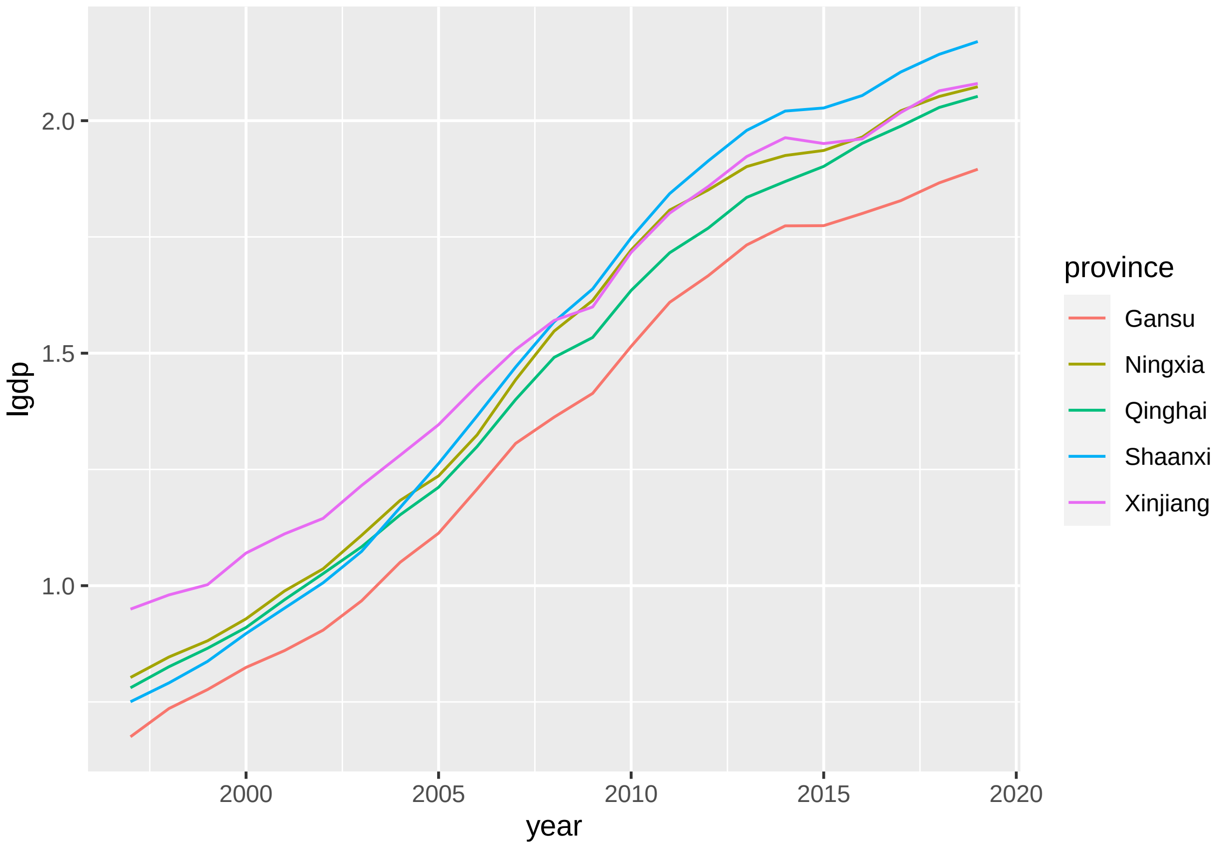

```
all_0915 %>%
  filter(region == "E") %>%
  ggplot(aes(x = year, y = lgdp, colour = province)) +
  geom_line()
```



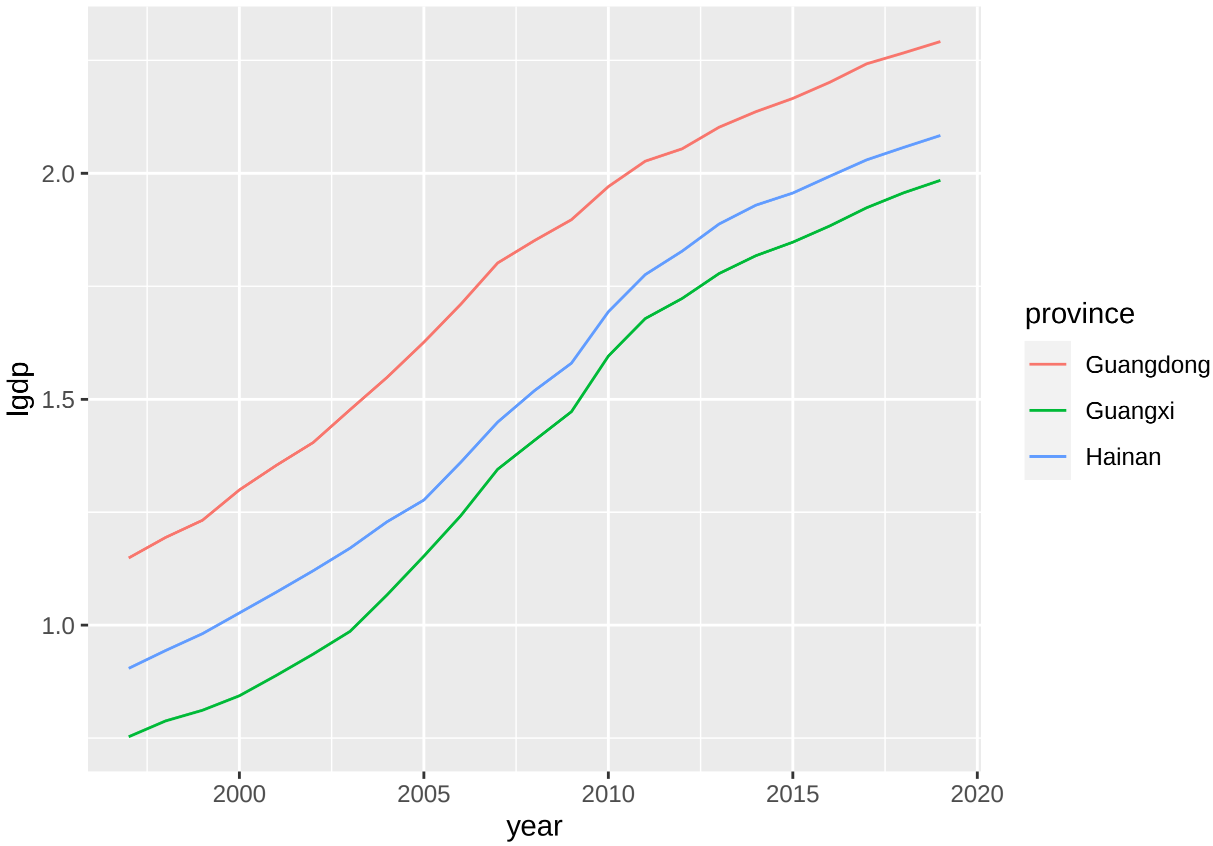

```
all_0915 %>%
  filter(region == "NE") %>%
  ggplot(aes(x = year, y = lgdp, colour = province)) +
  geom_line()
```



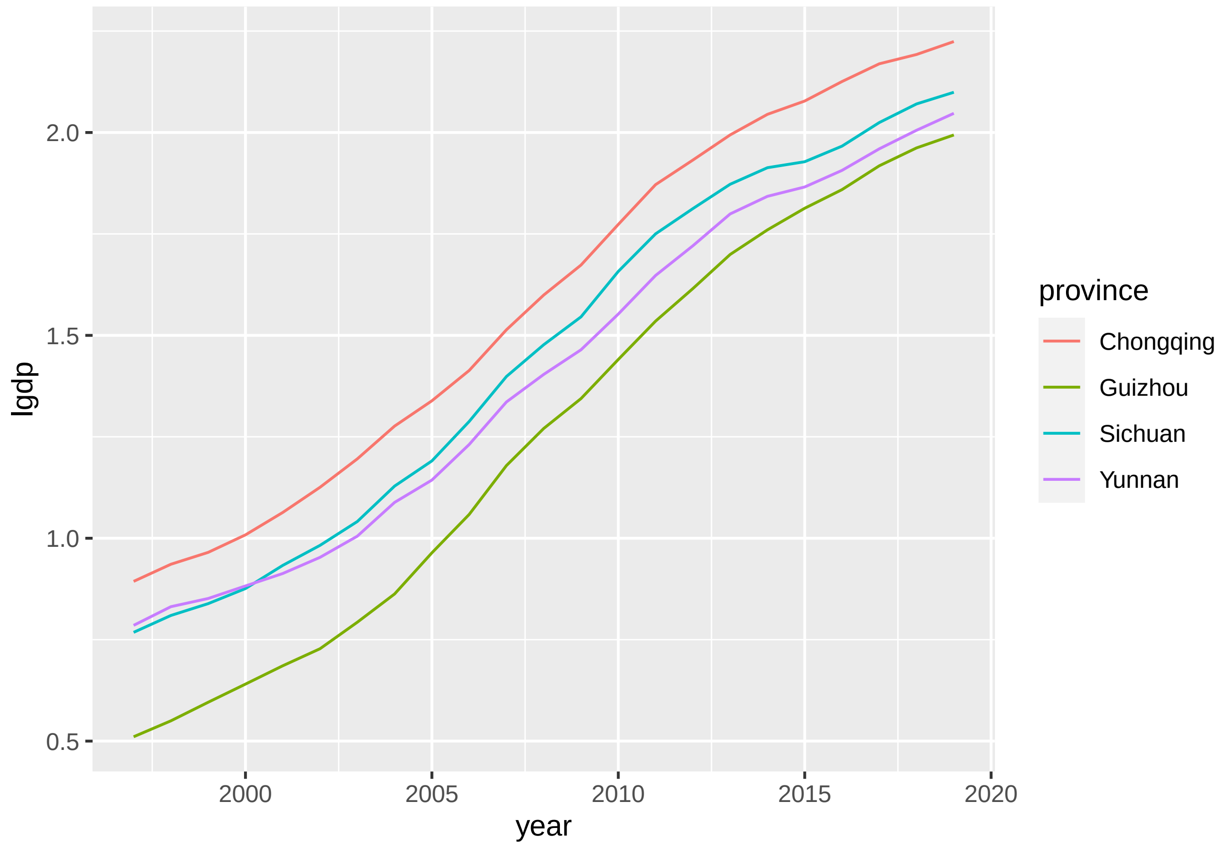

```
all_0915 %>%
  filter(region == "NW") %>%
  ggplot(aes(x = year, y = lgdp, colour = province)) +
  geom_line()
```




```r
all_0915 %>%
  filter(region == "S") %>%
  ggplot(aes(x = year, y = lgdp, colour = province)) +
  geom_line()
```

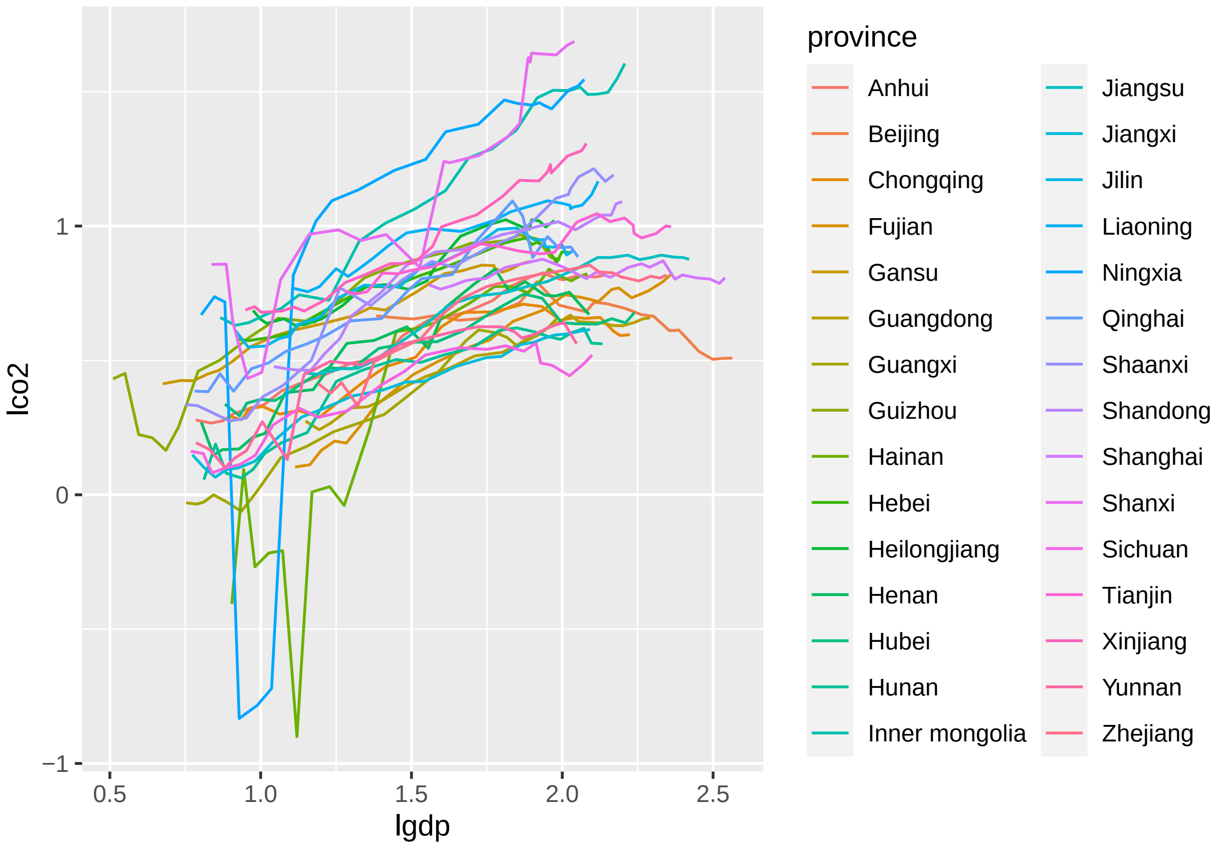



```r
all_0915 %>%
  filter(region == "SW") %>%
  ggplot(aes(x = year, y = lgdp, colour = province)) +
  geom_line()
```



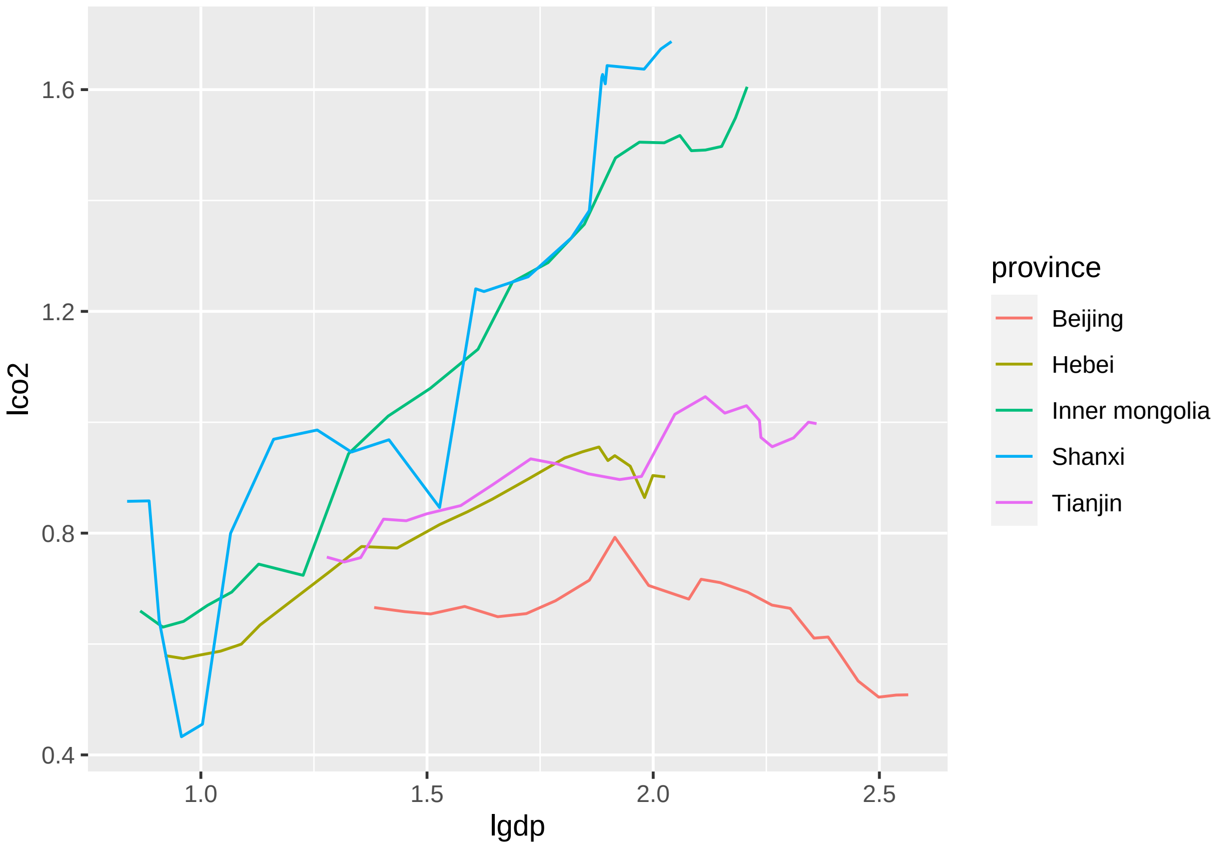
Plot lco2 against lgdp

```
all_0915 %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province, gg=TRUE)) +
  geom_line()
```

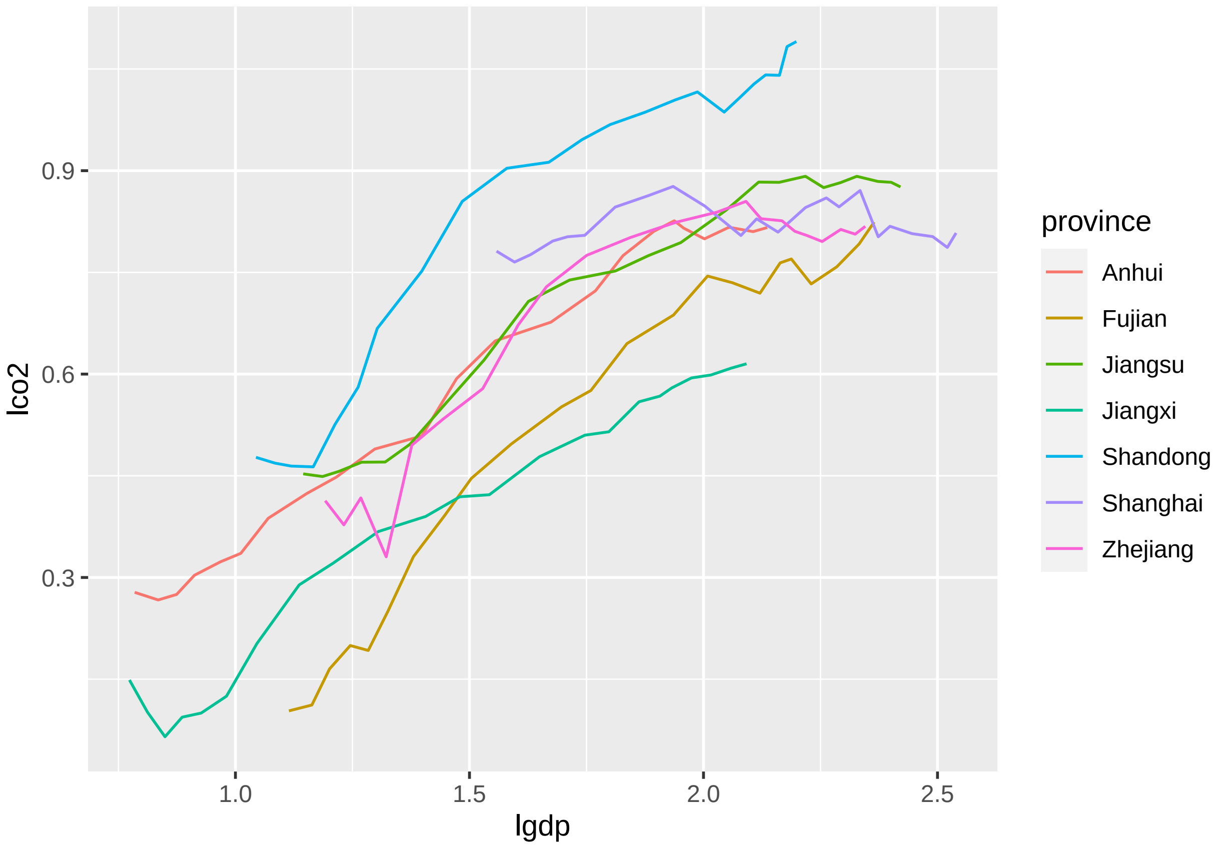



```
all_0915 %>%
  filter(region == "C") %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province, gg=TRUE)) +
  geom_line()
```

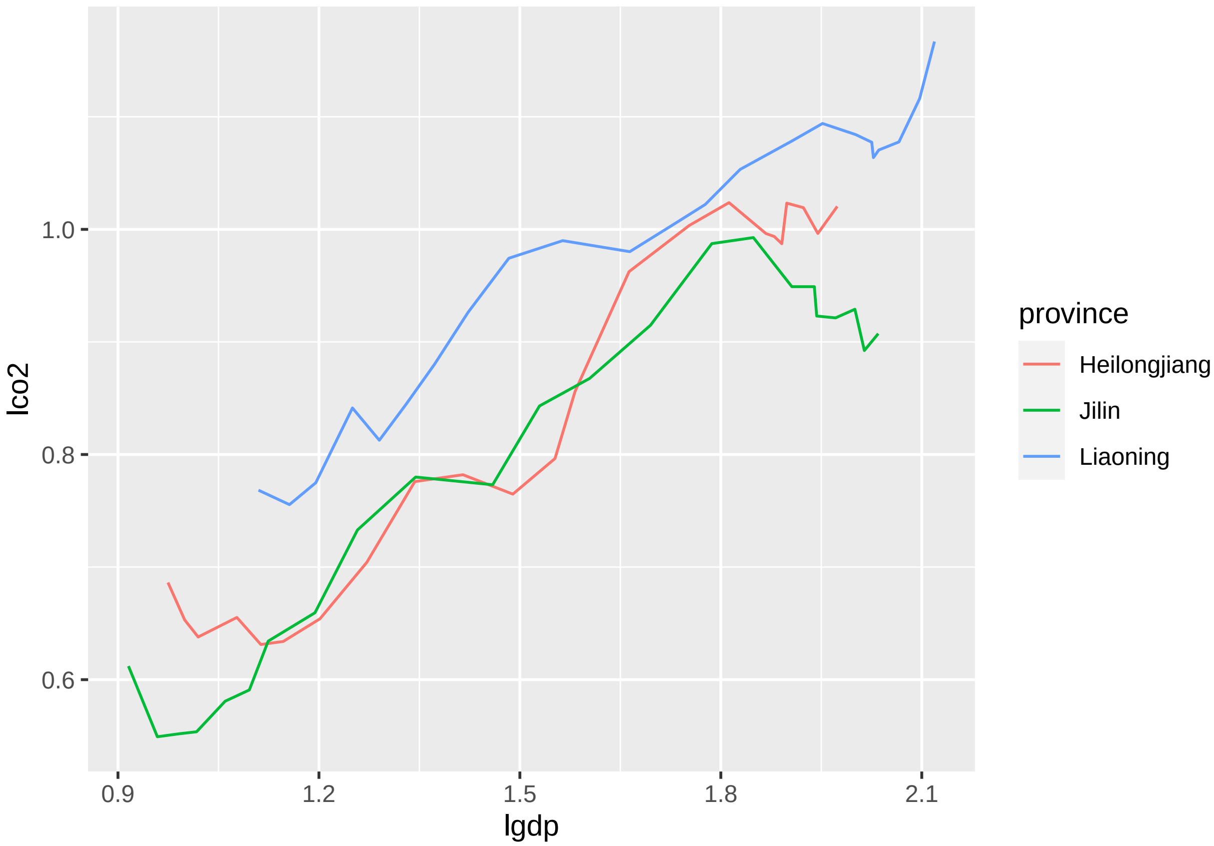

```
all_0915 %>%
  filter(region == "N") %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province)) +
  geom_line()
```



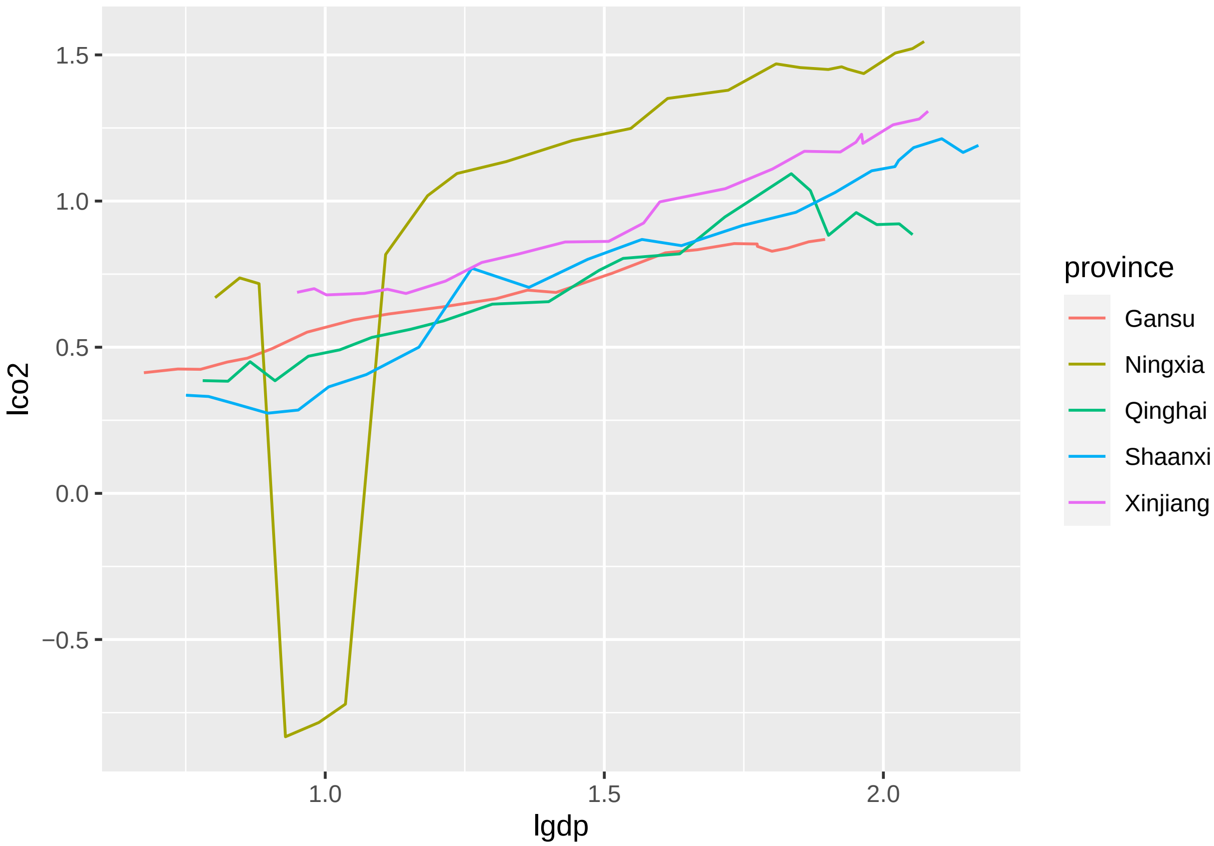

```
all_0915 %>%
  filter(region == "E") %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province)) +
  geom_line()
```



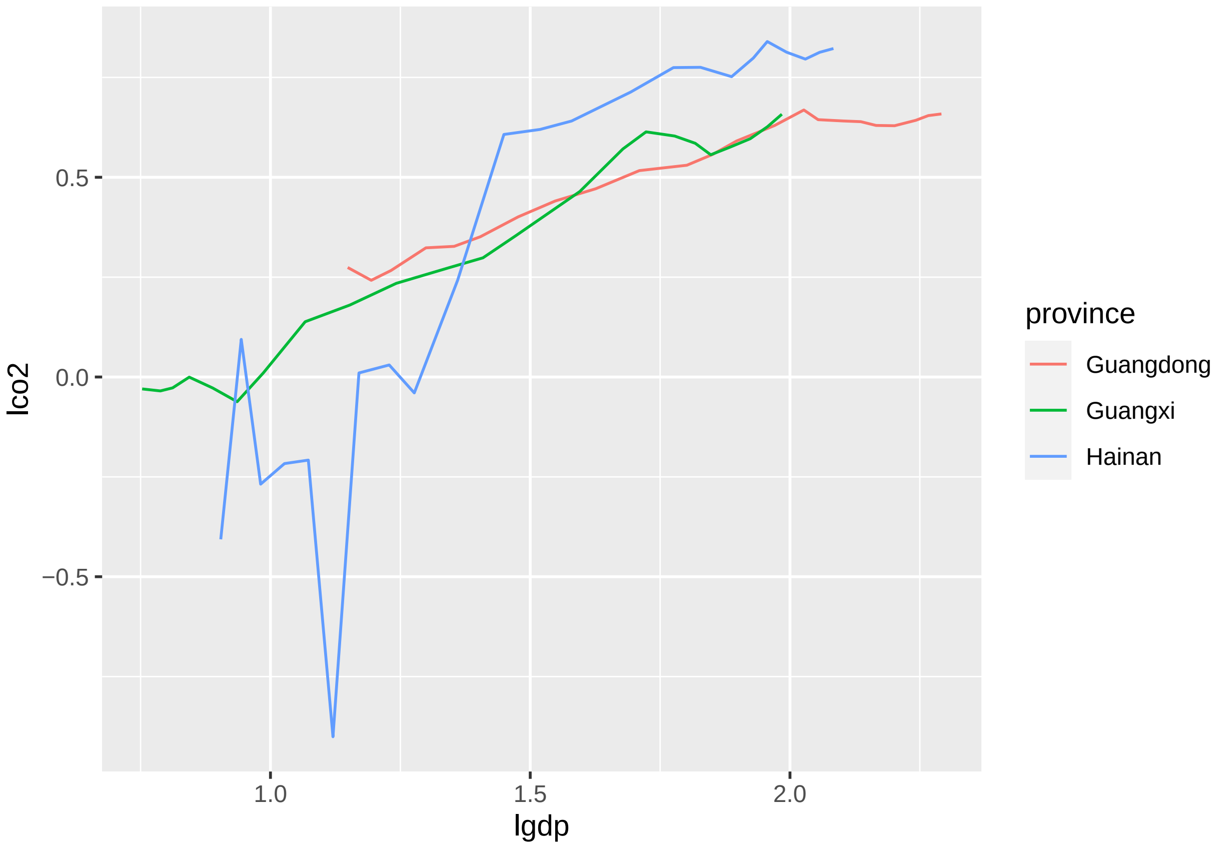

```
all_0915 %>%
  filter(region == "NE") %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province)) +
  geom_line()
```



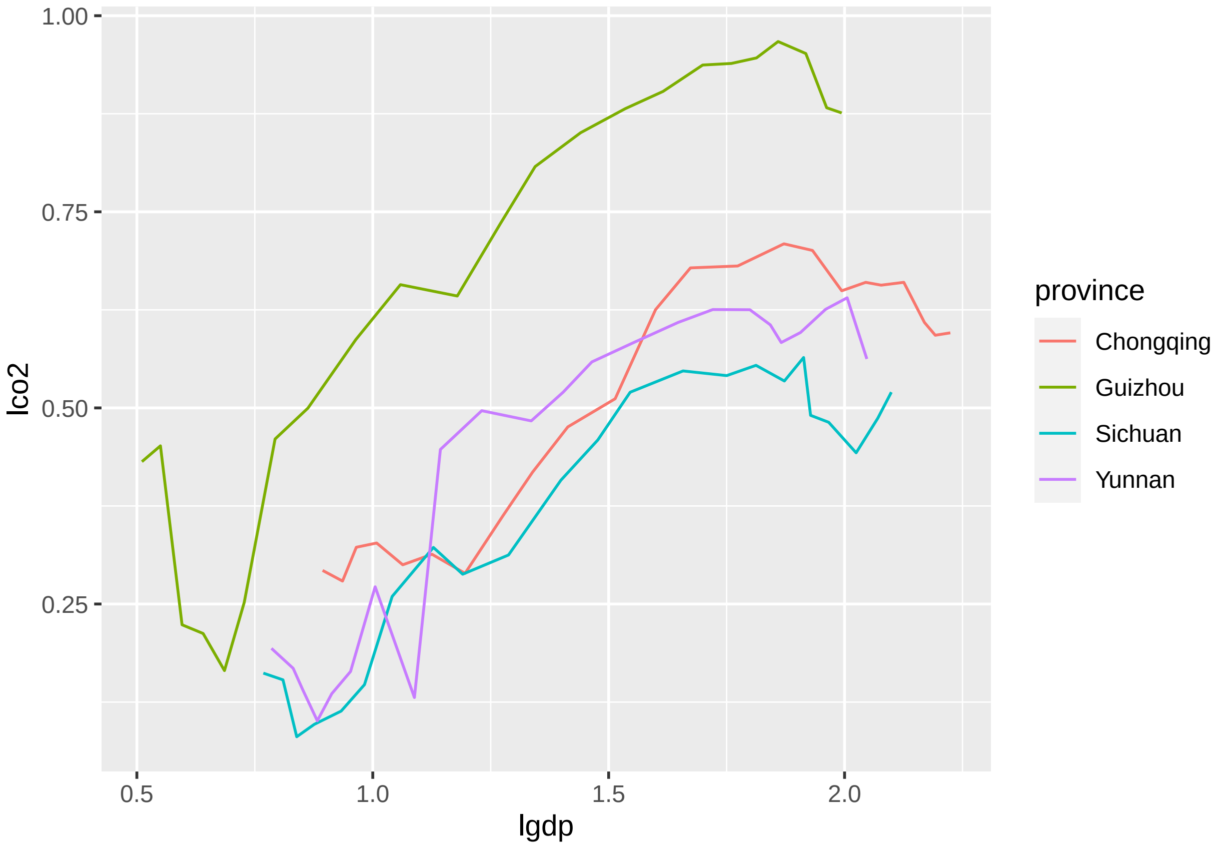

```r
all_0915 %>%
  filter(region == "NW") %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province)) +
  geom_line()
```




```
all_0915 %>%
  filter(region == "S") %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province)) +
  geom_line()
```

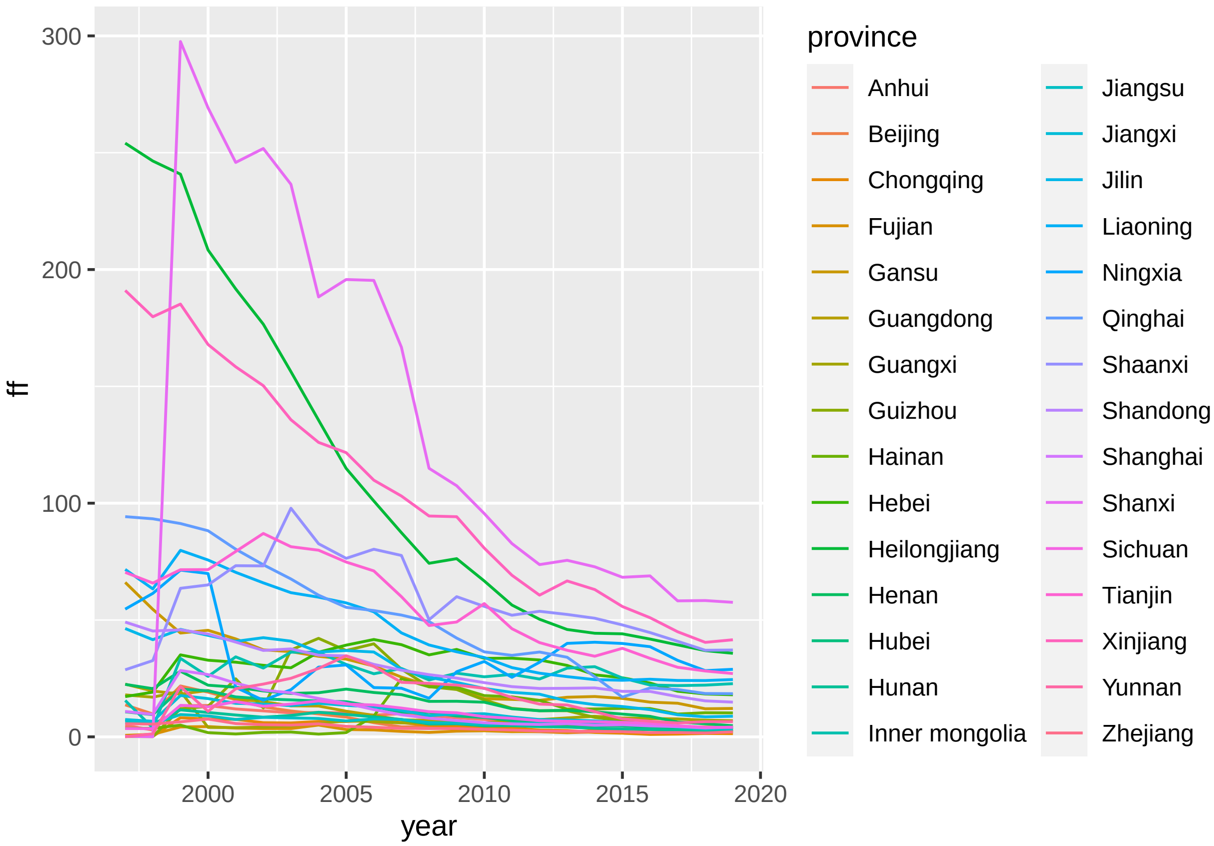



```
all_0915 %>%
  filter(region == "SW") %>%
  ggplot(aes(x = lgdp, y = lco2, colour = province)) +
  geom_line()
```

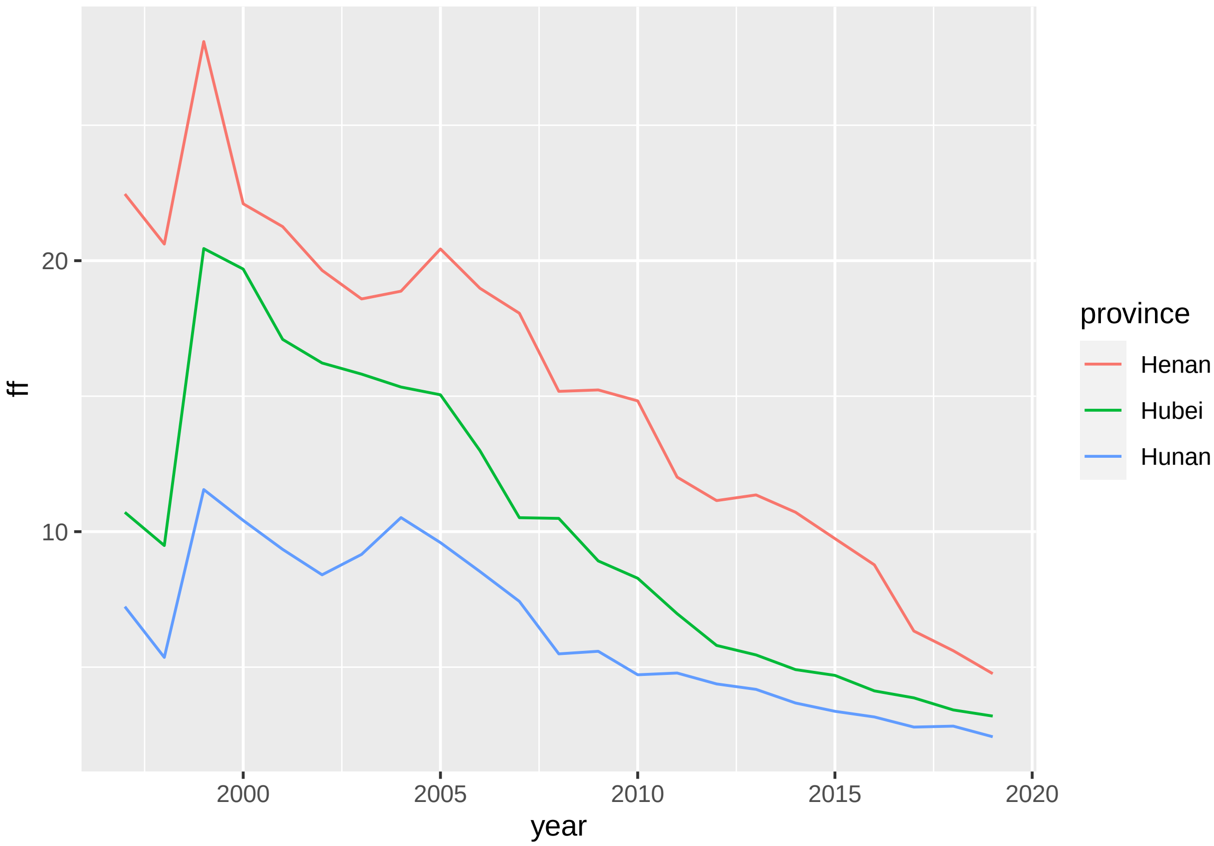


Plot ffindex against year

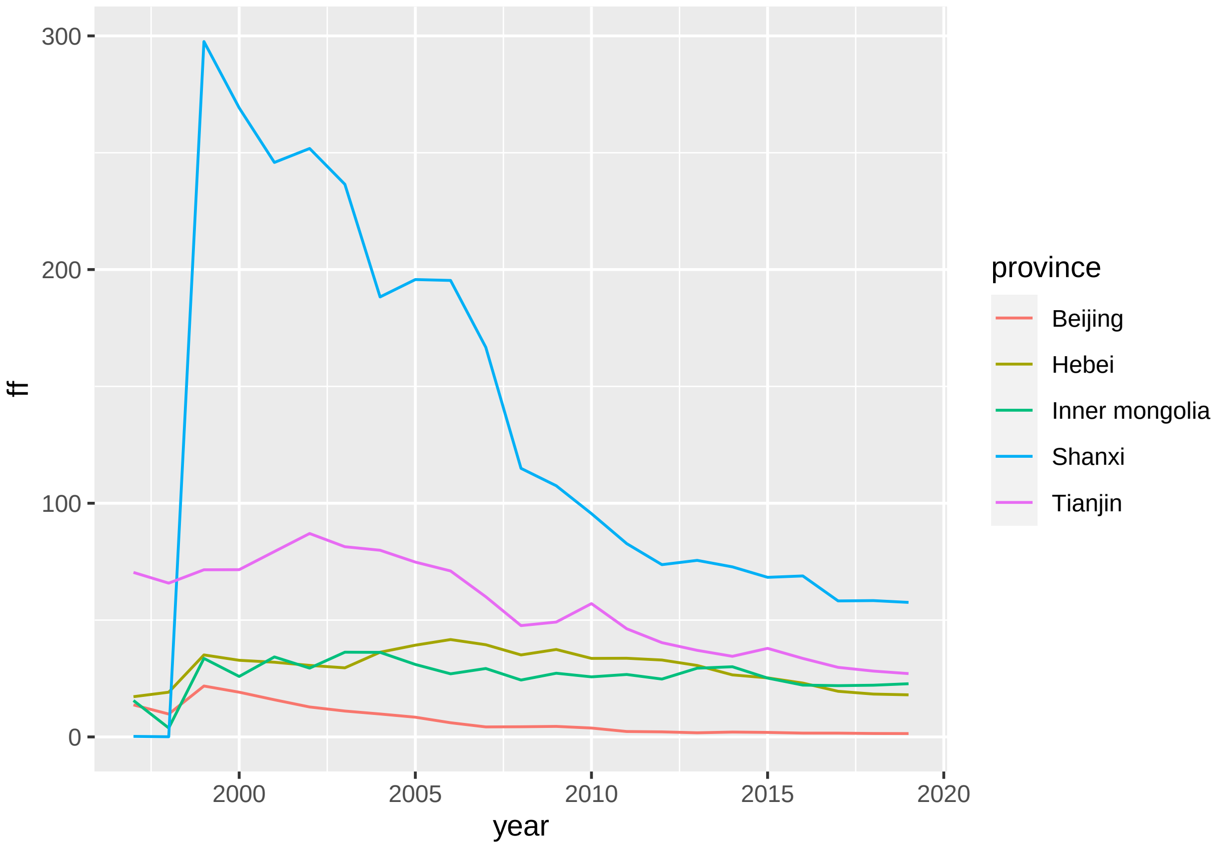
```
all_0915 %>%
  ggplot(aes(x = year, y = ff, colour = province, gg=TRUE)) +
  geom_line()
```



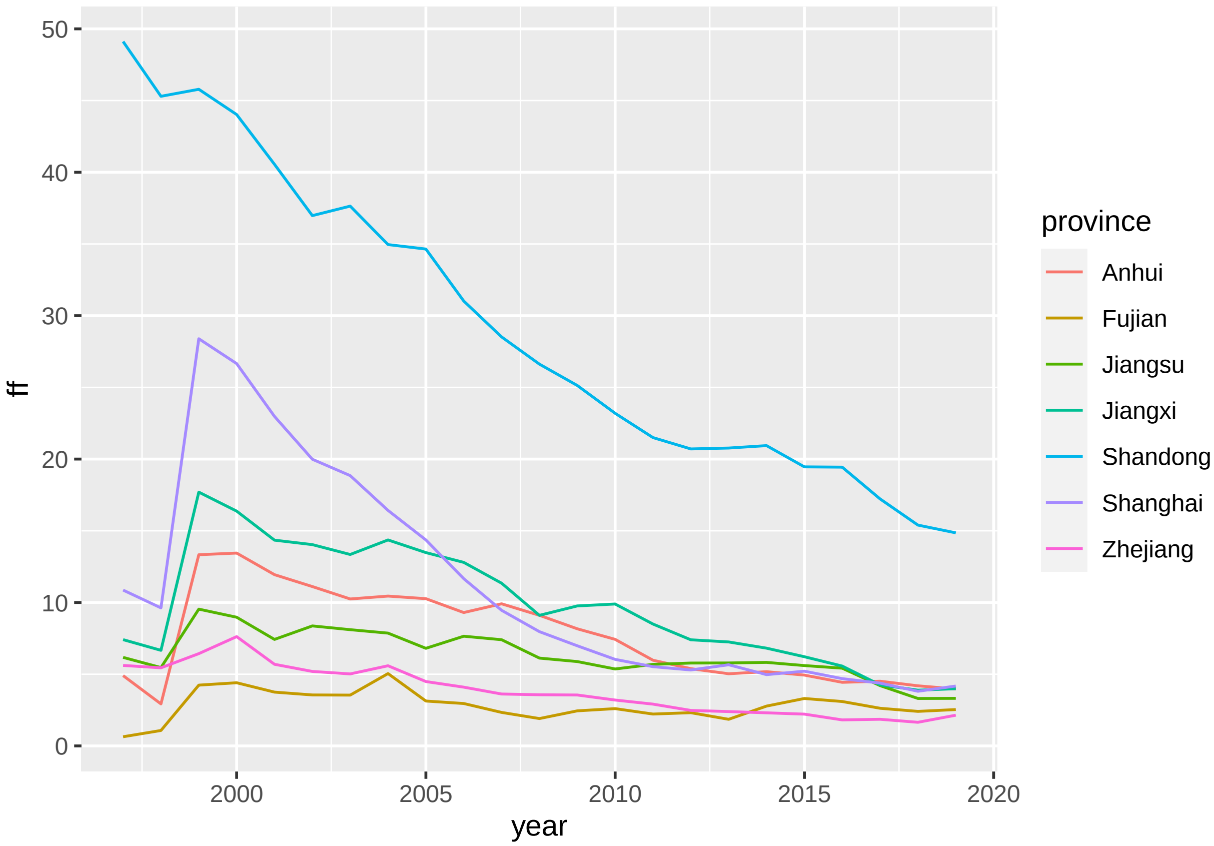

```r
all_0915 %>%
  filter(region == "C") %>%
  ggplot(aes(x = year, y = ff, colour = province)) +
  geom_line()
```



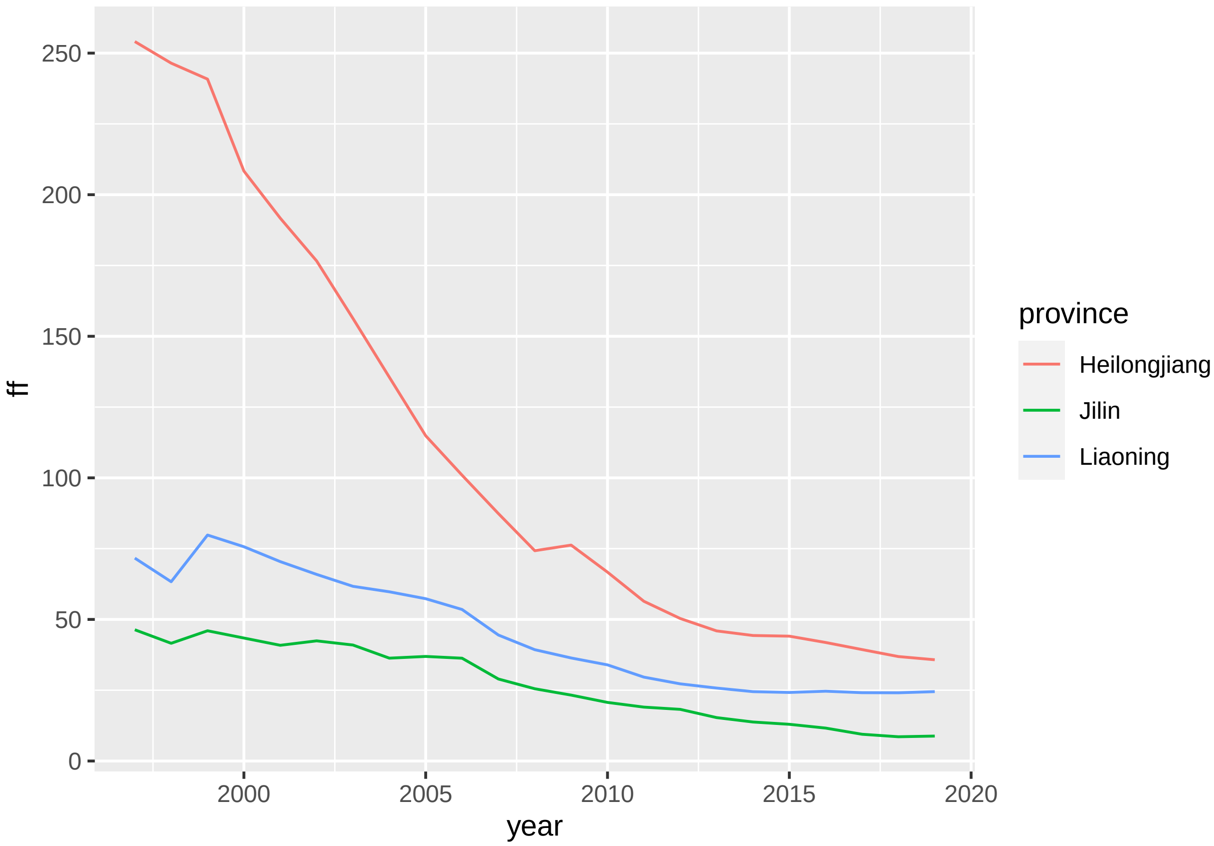

```r
all_0915 %>%
  filter(region == "N") %>%
  ggplot(aes(x = year, y = ff, colour = province)) +
  geom_line()
```



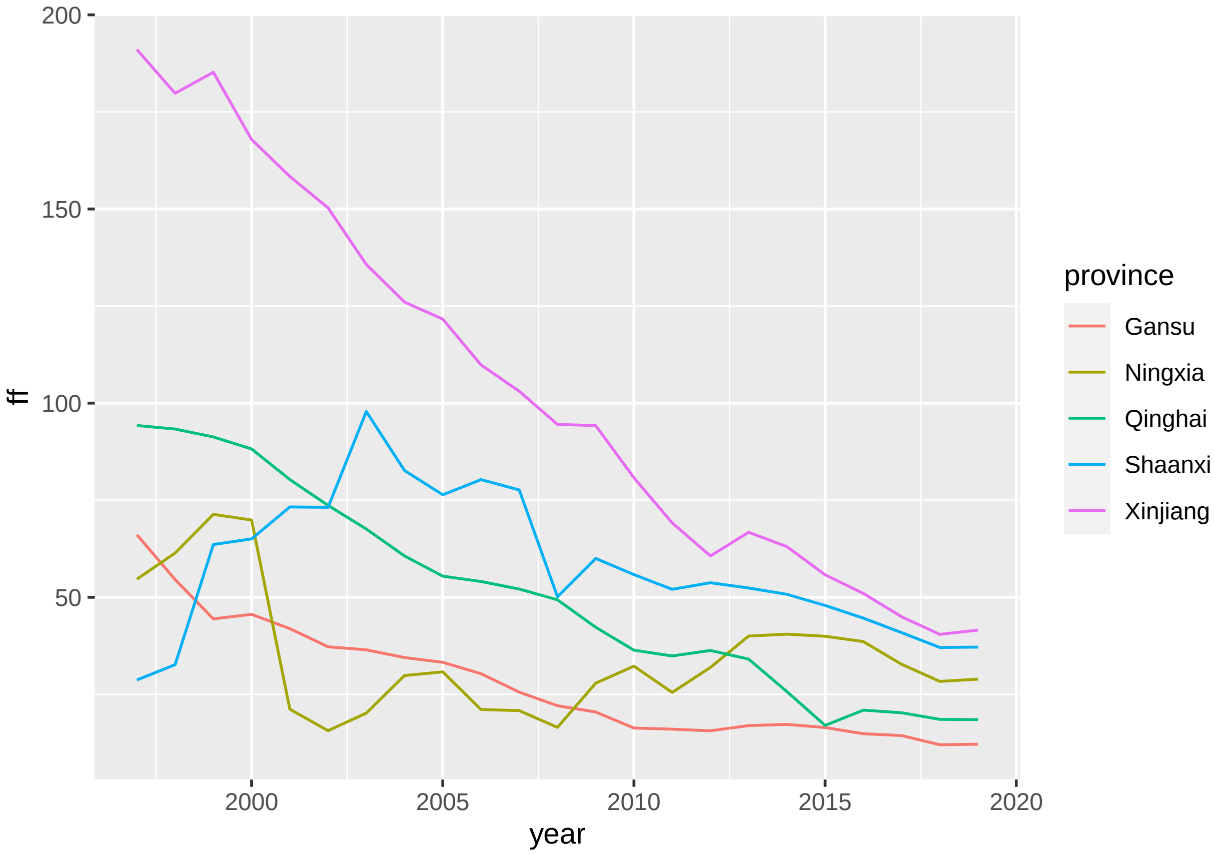

```
all_0915 %>%
  filter(region == "E") %>%
  ggplot(aes(x = year, y = ff, colour = province)) +
  geom_line()
```



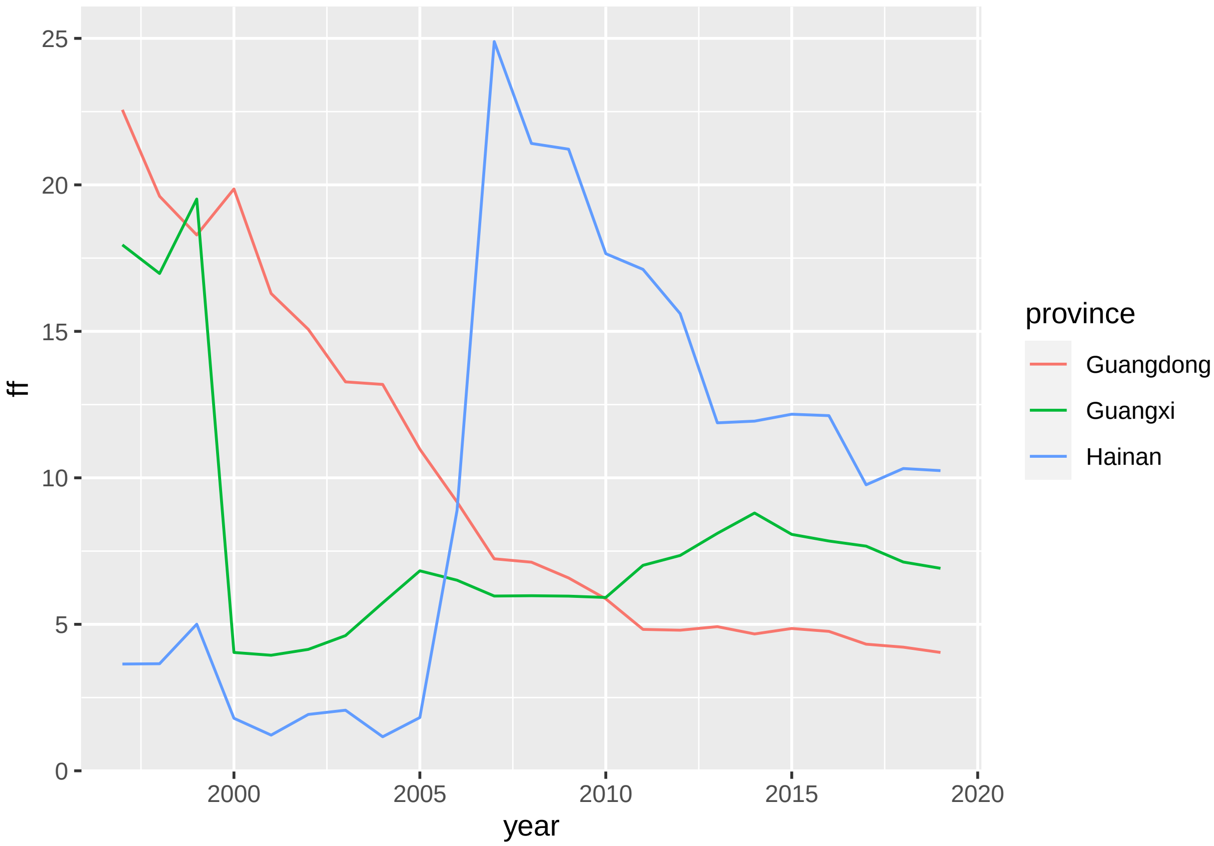

```
all_0915 %>%
  filter(region == "NE") %>%
  ggplot(aes(x = year, y = ff, colour = province)) +
  geom_line()
```



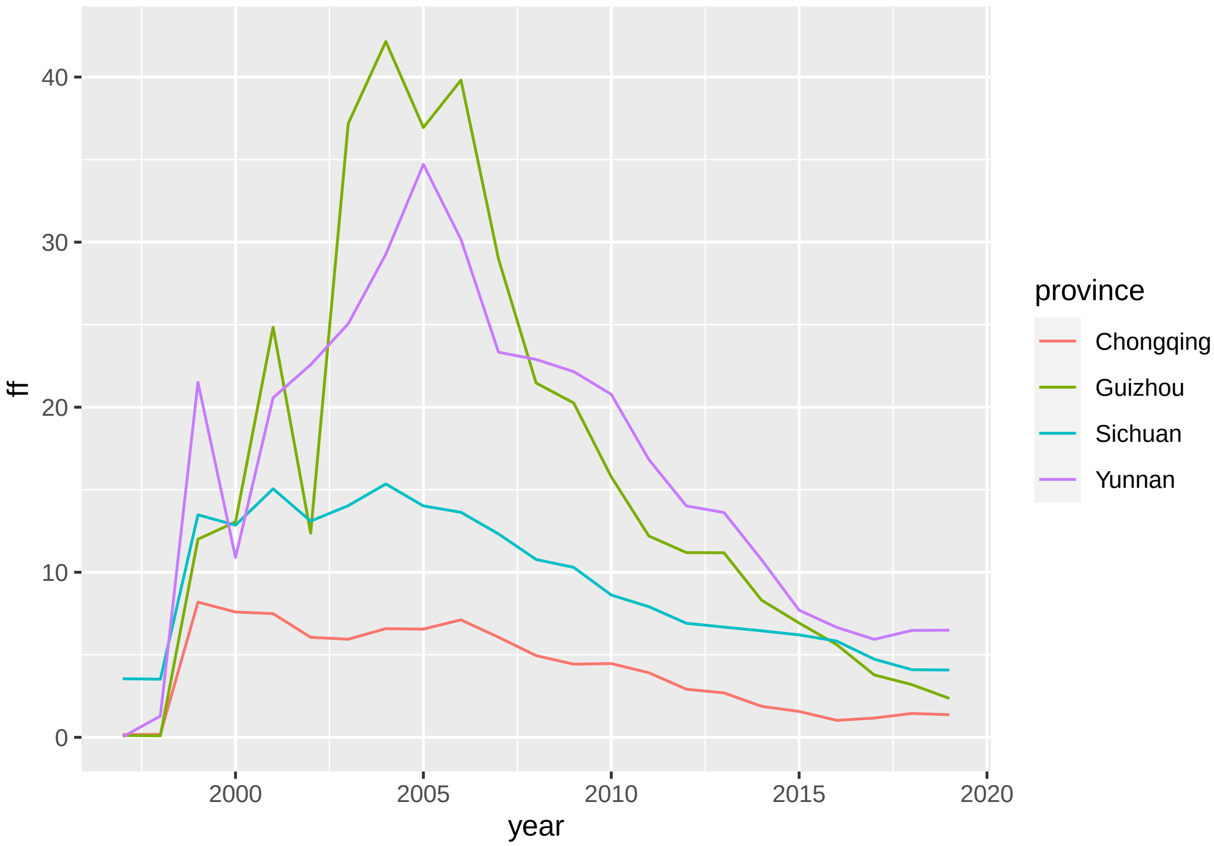

```r
all_0915 %>%
  filter(region == "NW") %>%
  ggplot(aes(x = year, y = ff, colour = province)) +
  geom_line()
```




```r
all_0915 %>%
  filter(region == "S") %>%
  ggplot(aes(x = year, y = ff, colour = province)) +
  geom_line()
```

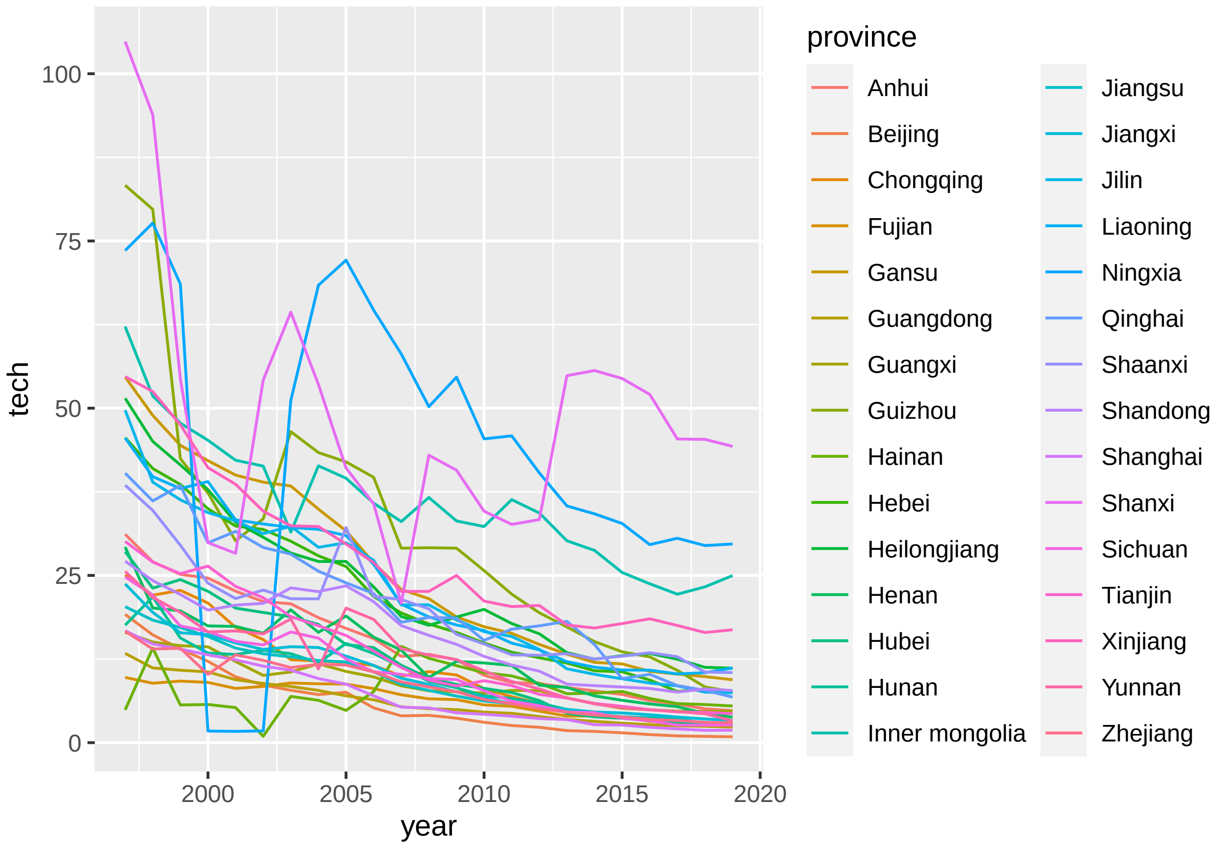



```
all_0915 %>%
  filter(region == "SW") %>%
  ggplot(aes(x = year, y = ff, colour = province)) +
  geom_line()
```

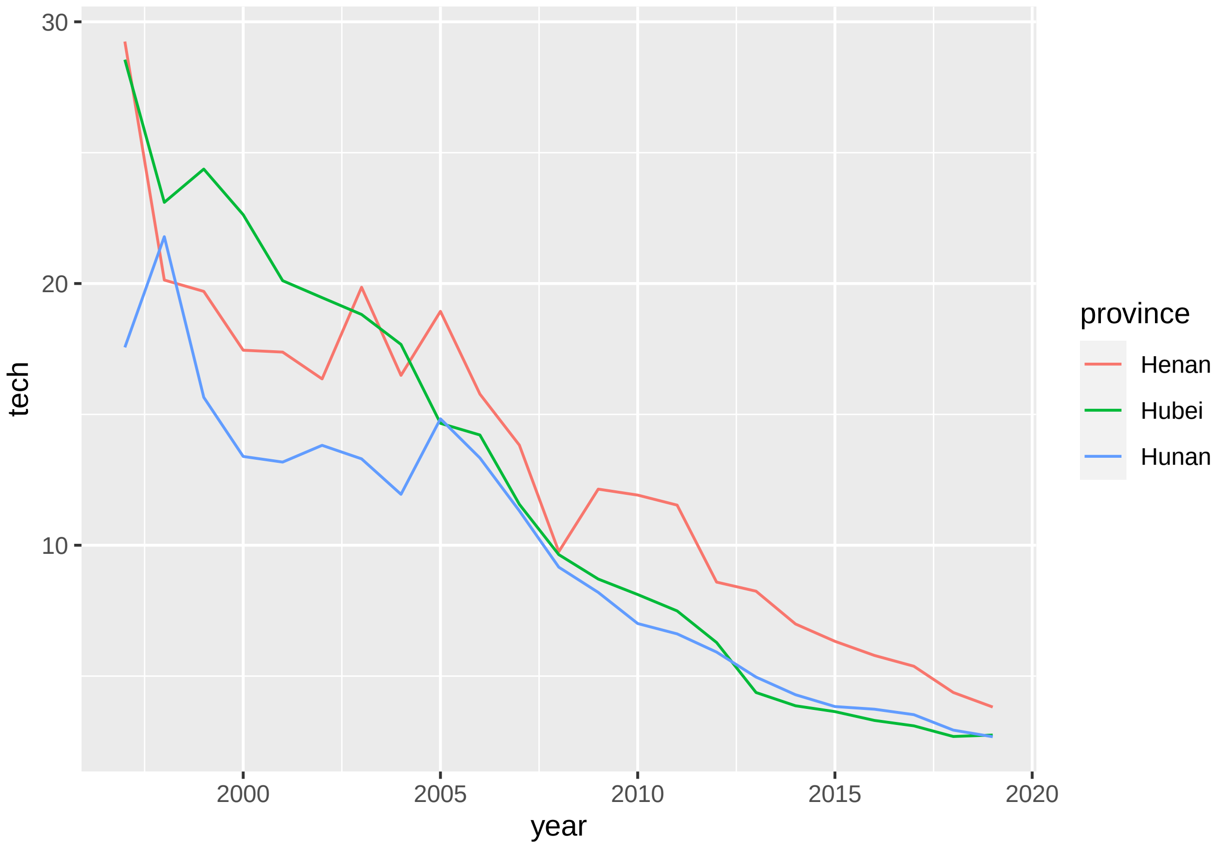


plot tech against year:

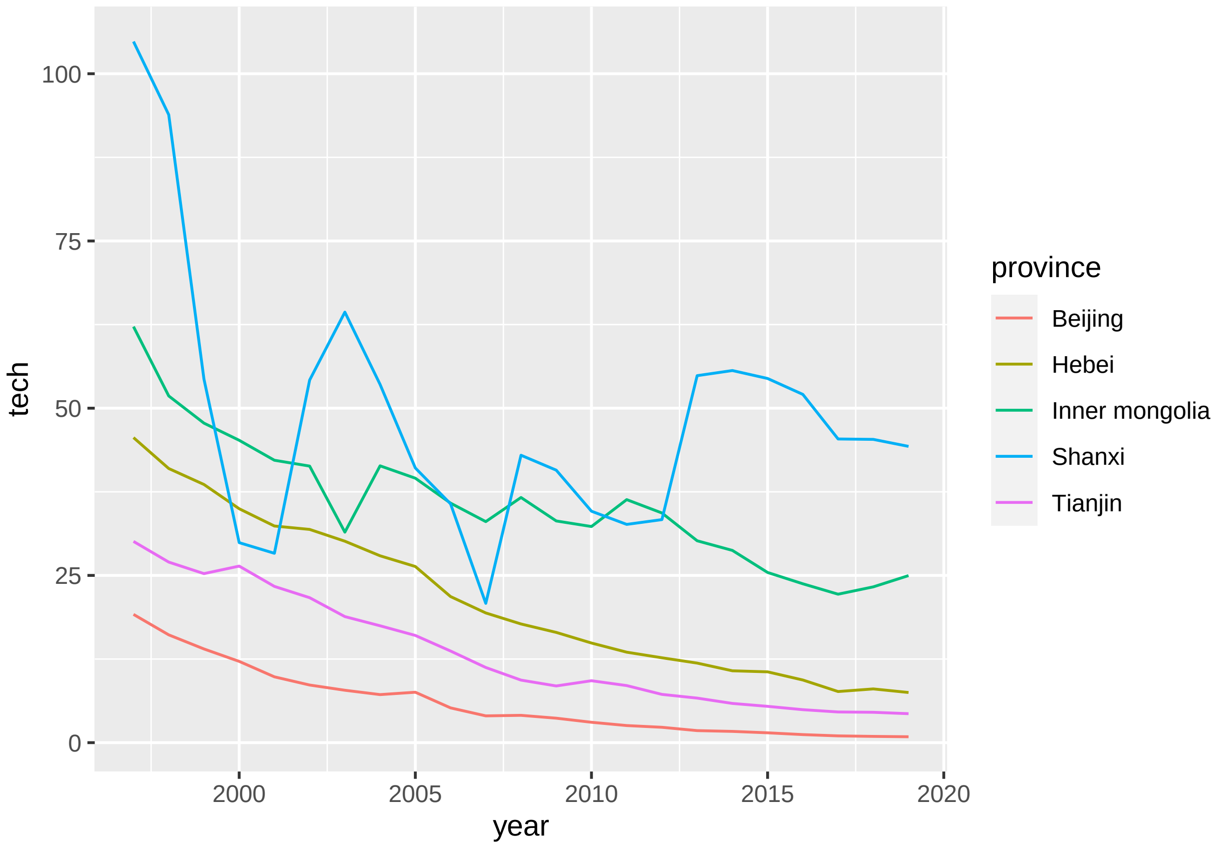
```
all_0915 %>%
  ggplot(aes(x = year, y = tech, colour = province, gg=TRUE)) +
  geom_line()
```



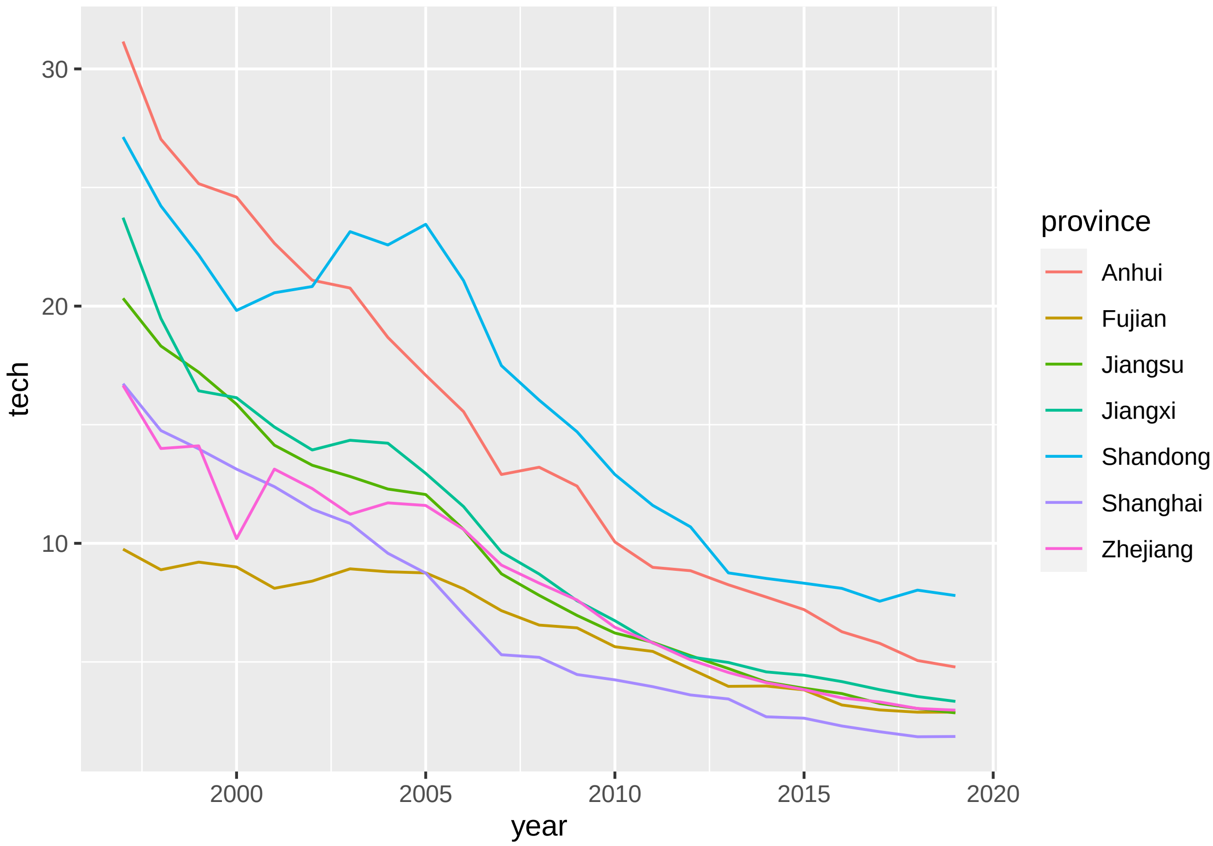

```
all_0915 %>%
  filter(region == "C") %>%
  ggplot(aes(x = year, y = tech, colour = province)) +
  geom_line()
```



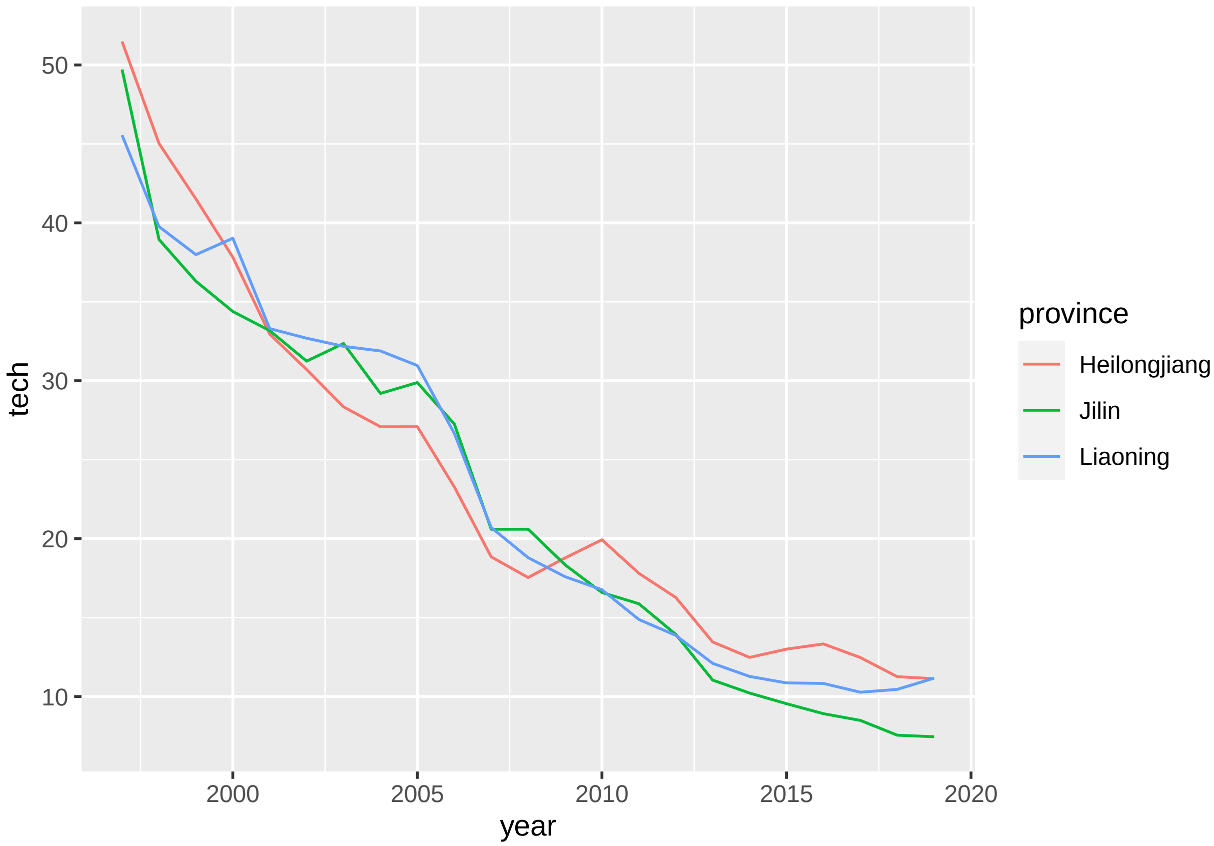

```
all_0915 %>%
  filter(region == "N") %>%
  ggplot(aes(x = year, y = tech, colour = province)) +
  geom_line()
```



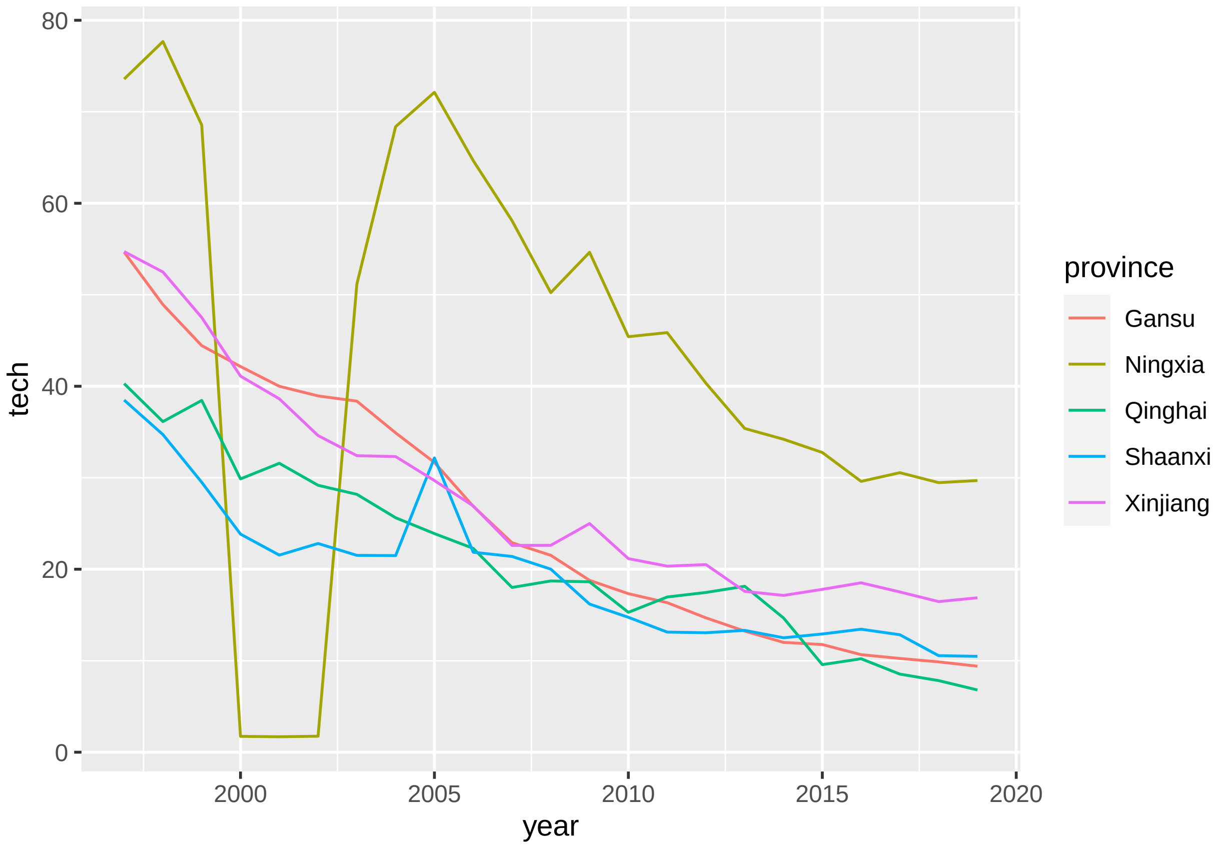

```
all_0915 %>%
  filter(region == "E") %>%
  ggplot(aes(x = year, y = tech, colour = province)) +
  geom_line()
```




```
all_0915 %>%
  filter(region == "NE") %>%
  ggplot(aes(x = year, y = tech, colour = province)) +
  geom_line()
```

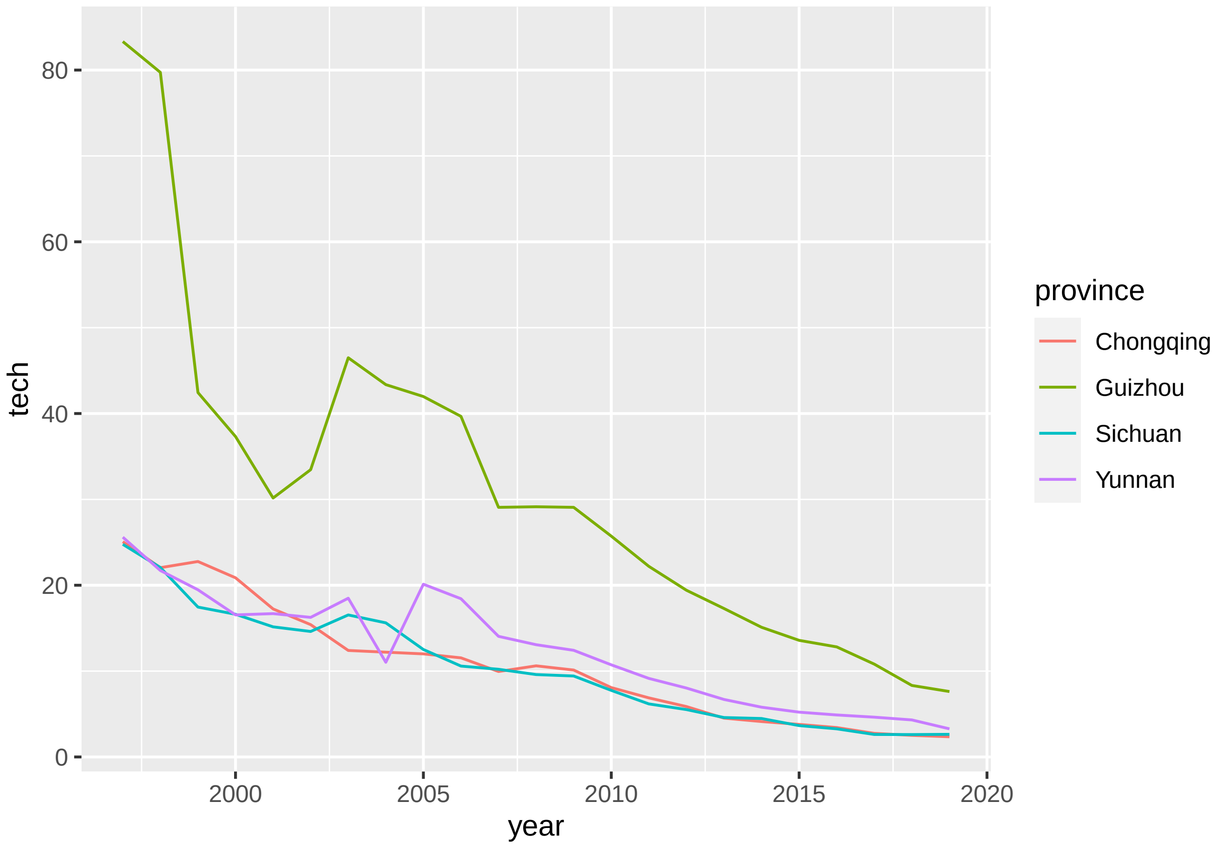



```r
all_0915 %>%
  filter(region == "NW") %>%
  ggplot(aes(x = year, y = tech, colour = province)) +
  geom_line()
```




```
all_0915 %>%
  filter(region == "S") %>%
  ggplot(aes(x = year, y = tech, colour = province)) +
  geom_line()
```


```
all_0915 %>%
  filter(region == "SW") %>%
  ggplot(aes(x = year, y = tech, colour = province)) +
  geom_line()
```



##Split the sample into training and testing sets: - data before 2016: to fit the model; - data from 2017 to 2019: to compare with prediction by the proposed model

```
ff<-all_0915$ff
tech<-all_0915$tech
a <- filter(all_0915, year<=2016)
```

Priliminary analysis

3D: Additive smoother

```
library(mgcv)
```

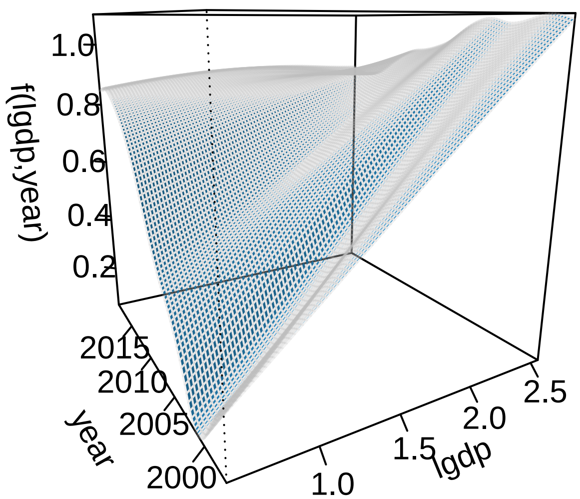
```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

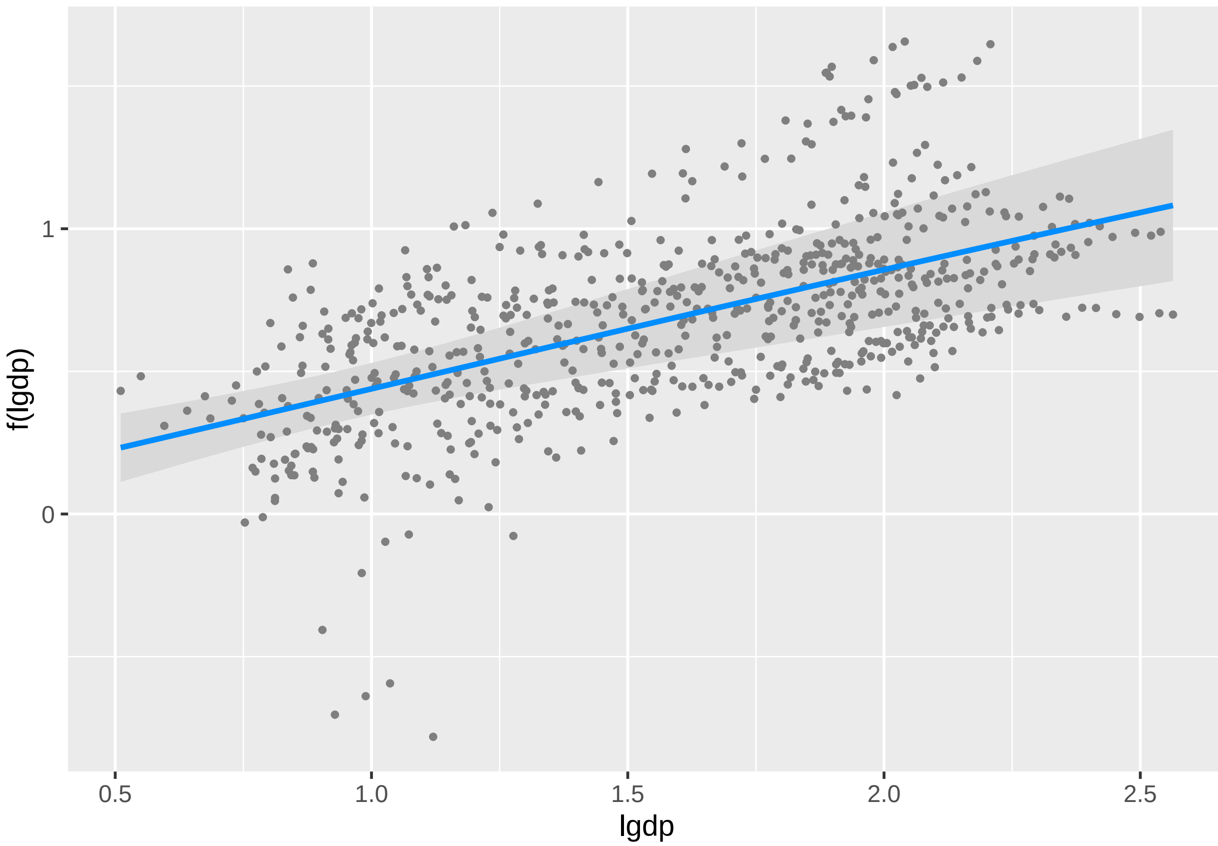
```
## The following object is masked from 'package:feasts':
##
##     ACF
```

```
## The following object is masked from 'package:forecast':
##
##     getResponse

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-40. For overview type 'help("mgcv-package")'.
```

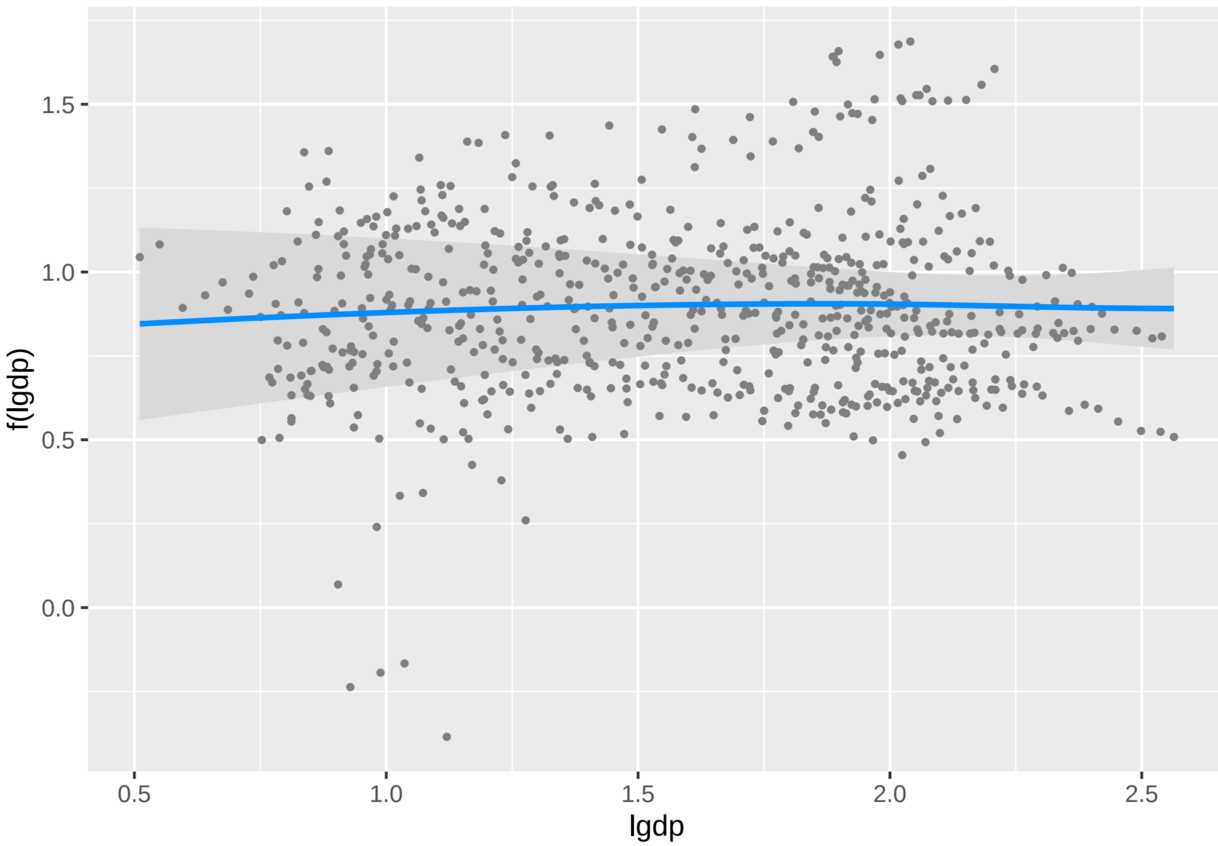
```
library(visreg)
fit4 <- gam(lco2~s(lgdp, year), data=all_0915)

visreg2d(fit4,"lgdp","year",  nn=99, plot.type="persp",
         ylab="\n\nyear", zlab="\n\nf(lgdp,year)")
```




```
visreg(fit4,"lgdp", cond=list(year=1997),  gg=TRUE)
```




```
visreg(fit4,"lgdp", cond=list(year=2019),  gg=TRUE)
```



FE and TVFE estimation for 7 regions

C (3)

```
e_0915 <- read_excel("~/Desktop/RP/all_0915.xlsx", sheet = "C", range = "C1:K70")
# name the vectors:
colnames(e_0915) <- c("province","year","lco2","lgdp","lgdp2","ff","tech","region","yearstd")
e <- filter(e_0915, year<=2016)
ff<-e$ff
tech<-e$tech
```

FE

```
mod.fe <- plm::plm(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"), model="within", data=
summary(mod.fe)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm::plm(formula = lco2 ~ lgdp + lgdp2 + ff + tech + yearstd,
##     data = e, model = "within", index = c("province", "year"))
```

```
##
## Balanced Panel: n = 3, T = 20, N = 60
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.0540863 -0.0213011  0.0012285  0.0142195  0.0628514
##
## Coefficients:
##         Estimate Std. Error  t-value  Pr(>|t|)
## lgdp   1.9514522  0.0887706  21.9831 < 2.2e-16 ***
## lgdp2 -0.4216696  0.0308418 -13.6720 < 2.2e-16 ***
## ff    -0.0047632  0.0016944  -2.8112  0.006902 **
## tech   0.0232947  0.0016907  13.7779 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    2.4709
## Residual Sum of Squares: 0.043977
## R-Squared:      0.9822
## Adj. R-Squared: 0.98019
## F-statistic: 731.217 on 4 and 53 DF, p-value: < 2.22e-16
```

```
mod.fe.CI <- confint(mod.fe, level = 0.68)
mod.fe.CI
```

```
##               16 %         84 %
## lgdp   1.863173615  2.039730809
## lgdp2 -0.452340451 -0.390998700
## ff    -0.006448215 -0.003078276
## tech   0.021613311  0.024976018
```

TVFE

Kernel: Gaussian Bandwidth: 0.6 Method: within

```
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                  data = e, method ="within", bw =NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.6004303
```

```
# Bootstrapping to get 95% confidence intervals
mod.tvfe.CI <- confint(mod.tvfe, level = 0.68)
mod.tvfe.CI
```

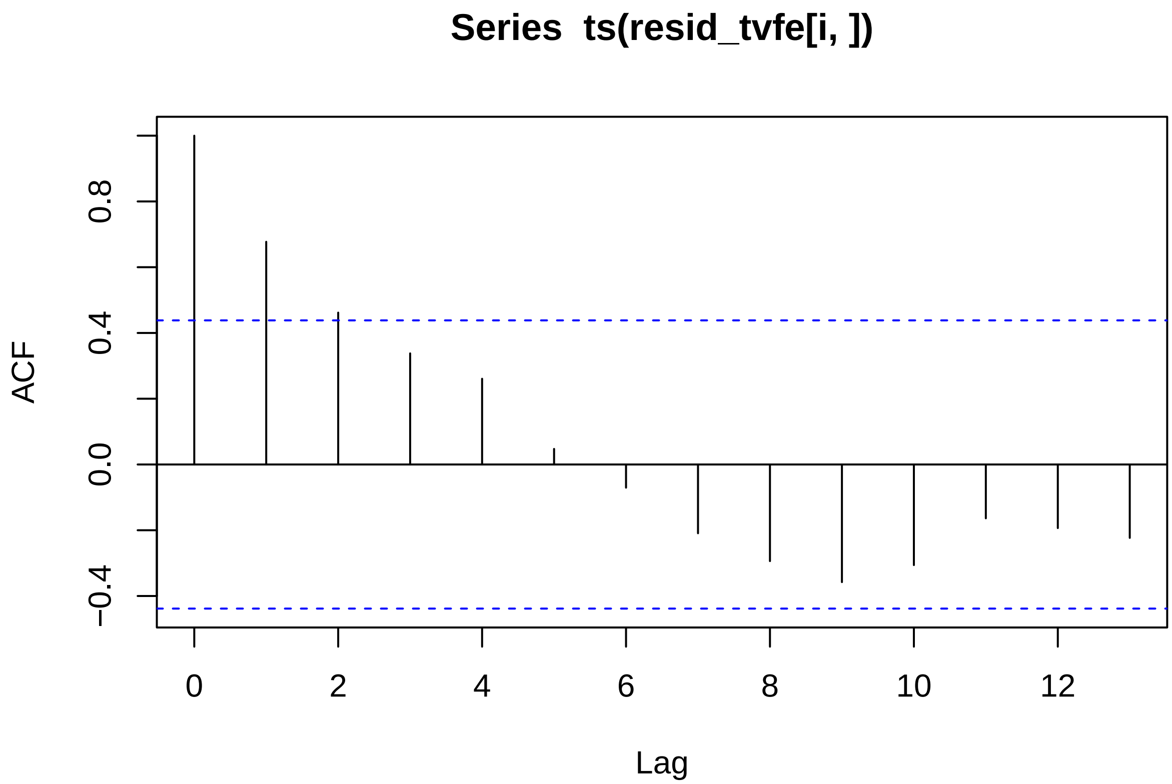
```
##
## Class:  tvplm
##
## Mean of coefficient estimates:
## ================================
##      lgdp     lgdp2        ff      tech   yearstd
##  1.943222 -0.415187 -0.004775  0.023893 -1.633792
##
```

```
## LOWER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  1.803628 -0.468324 -0.007892  0.020764 -1.737299
##
## UPPER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  2.082815 -0.362049 -0.001658  0.027022 -1.530284
##
## Bandwidth:  0.6004
```

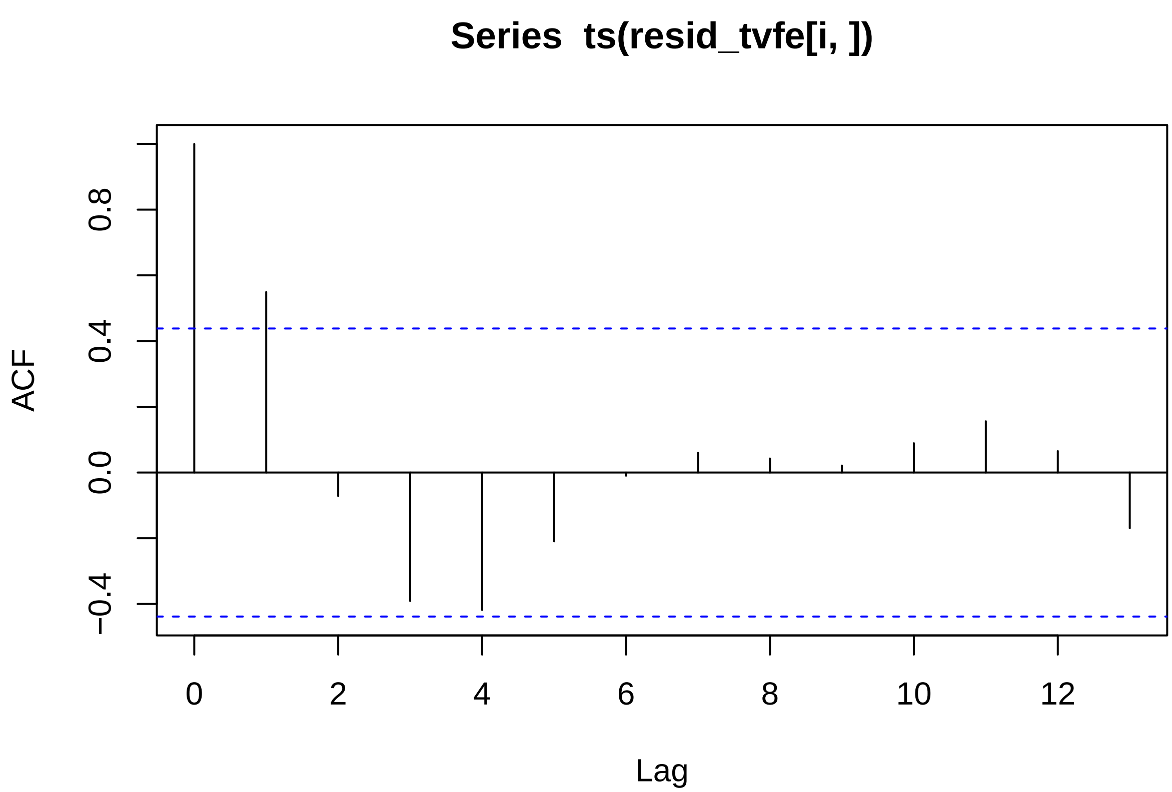
```
residtvfe <- mod.tvfe$residuals
residtvfe
```

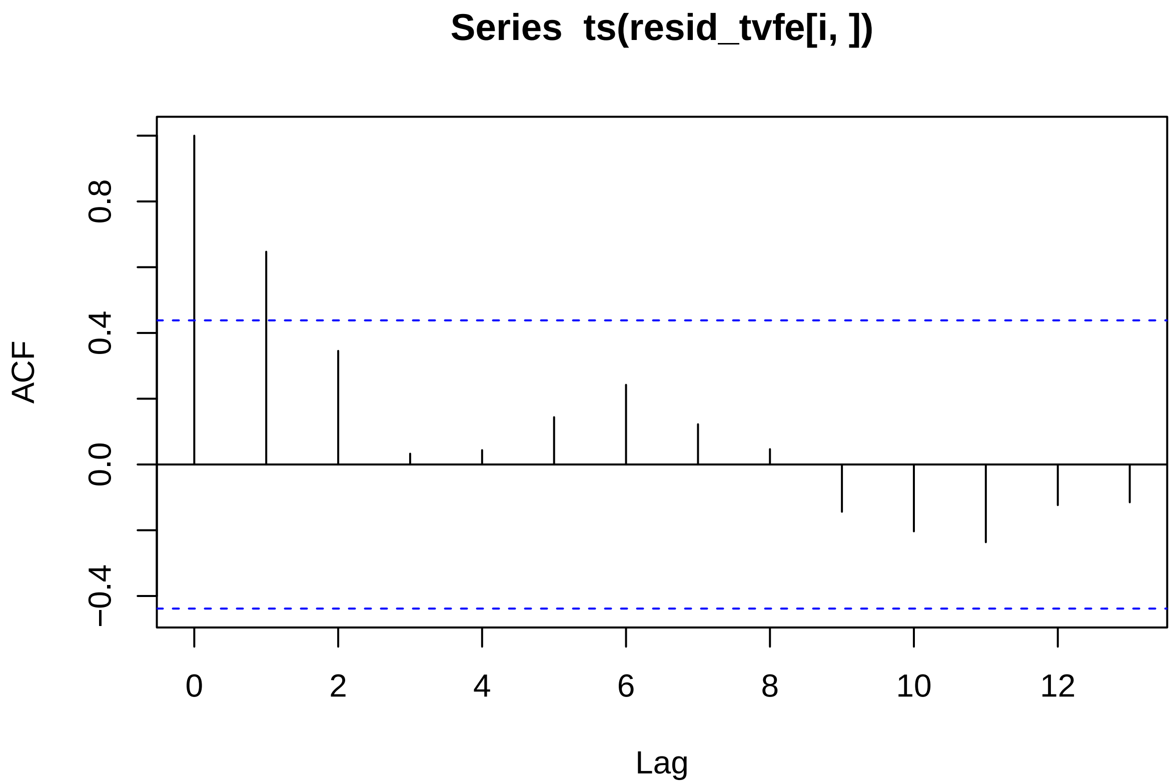
```
##  [1]  0.0115350687  0.0423870009  0.0636118846  0.0344204403  0.0267938081
##  [6]  0.0150321380  0.0104533836  0.0101744747  0.0292572545  0.0333062540
## [11]  0.0450719132  0.0004259674  0.0598066360  0.0977387699  0.1129089082
## [16]  0.0786065412  0.1033150035  0.0862396359  0.0709936770  0.0653575245
## [21] -0.0351624687 -0.0161116091  0.0111124743  0.0115624744  0.0055740265
## [26]  0.0012997501 -0.0003862680  0.0006187770 -0.0027596994  0.0008038279
## [31] -0.0087346027 -0.0146079669 -0.0101717517  0.0242152033  0.0518447884
## [36]  0.0377587244 -0.0274515328 -0.0327500712 -0.0336533095 -0.0358193800
## [41]  0.0052888945 -0.0116758529 -0.0021913595 -0.0299313484 -0.0409216472
## [46] -0.0428856484 -0.0486283769 -0.0593343194 -0.0319636692 -0.0367698582
## [51] -0.0427046033 -0.0696896199 -0.0679494038 -0.0712458675 -0.0407313318
## [56] -0.0417863868 -0.0571807679 -0.0759540059 -0.0869016065 -0.0697978758
```

```
N=3
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```




```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value

## Warning in kpss.test(ts(data_i)): p-value greater than printed p-value
```





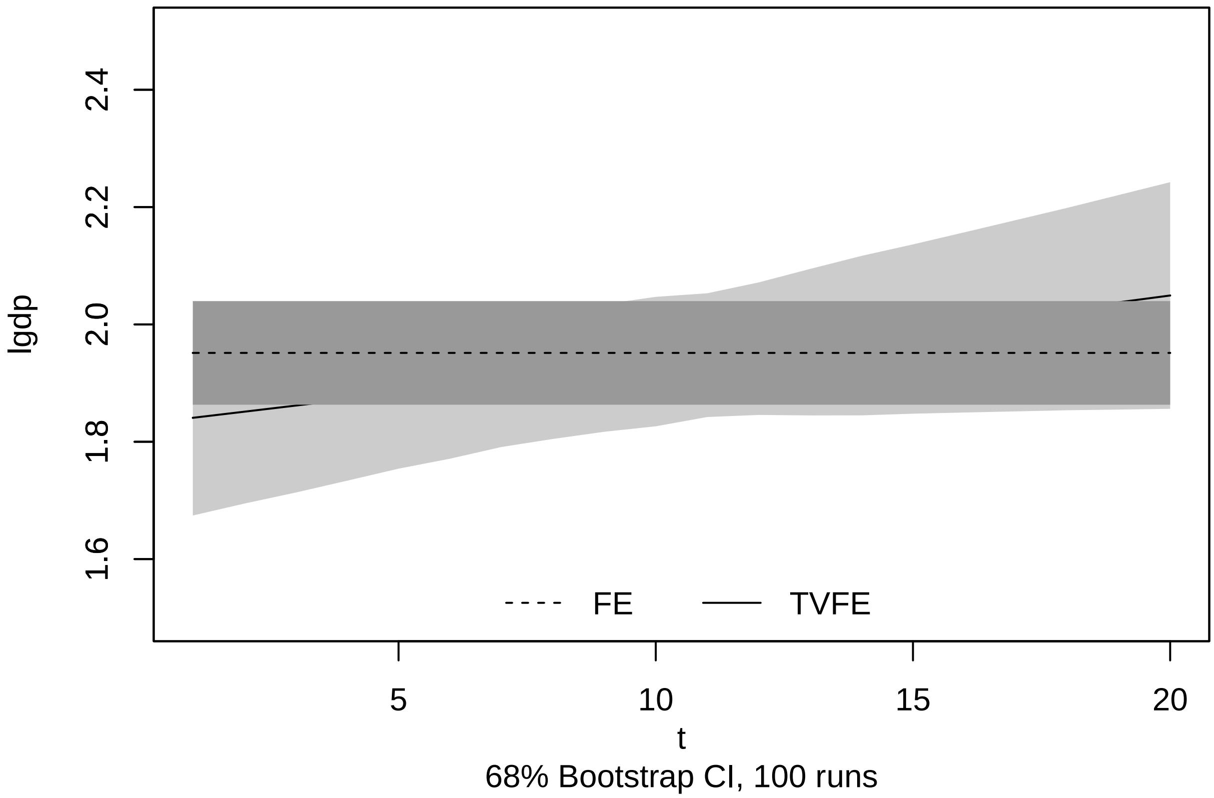

```
pro_reject
```

```
##  [1]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

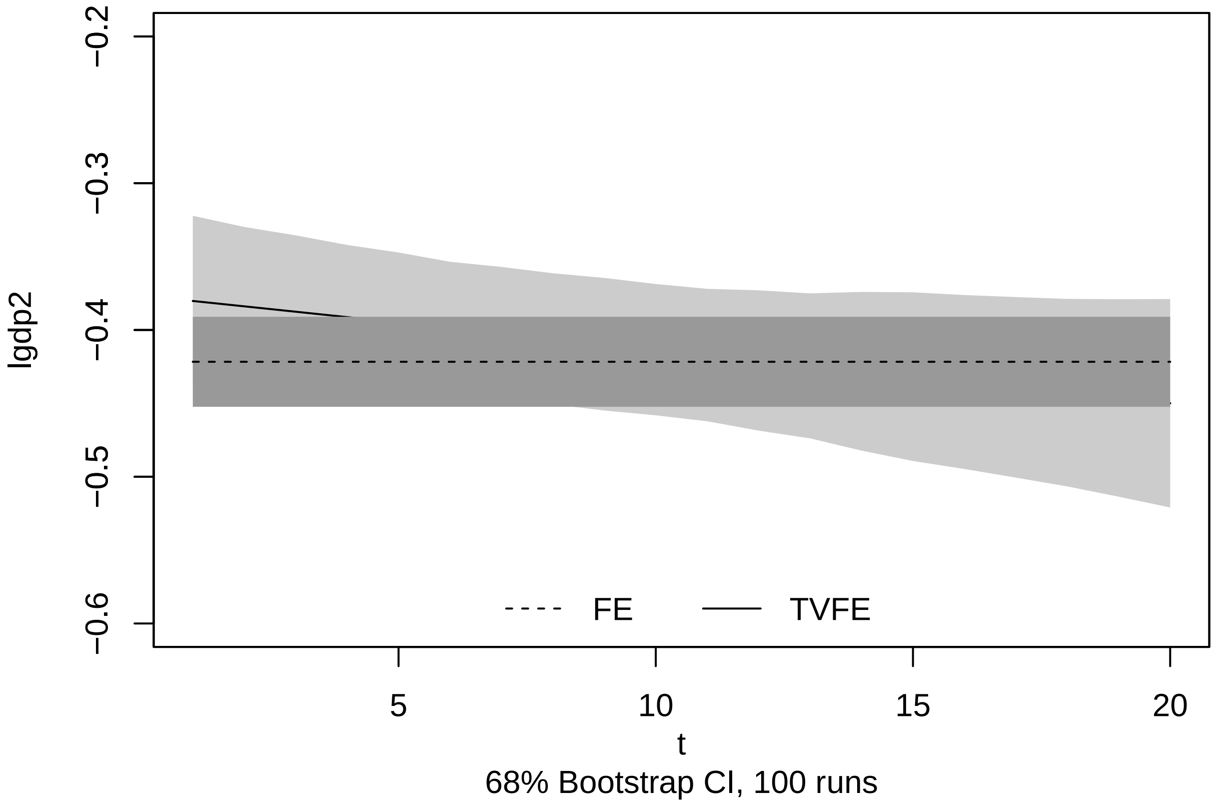
Post-estimation

1) Comparison plot of the within estimators

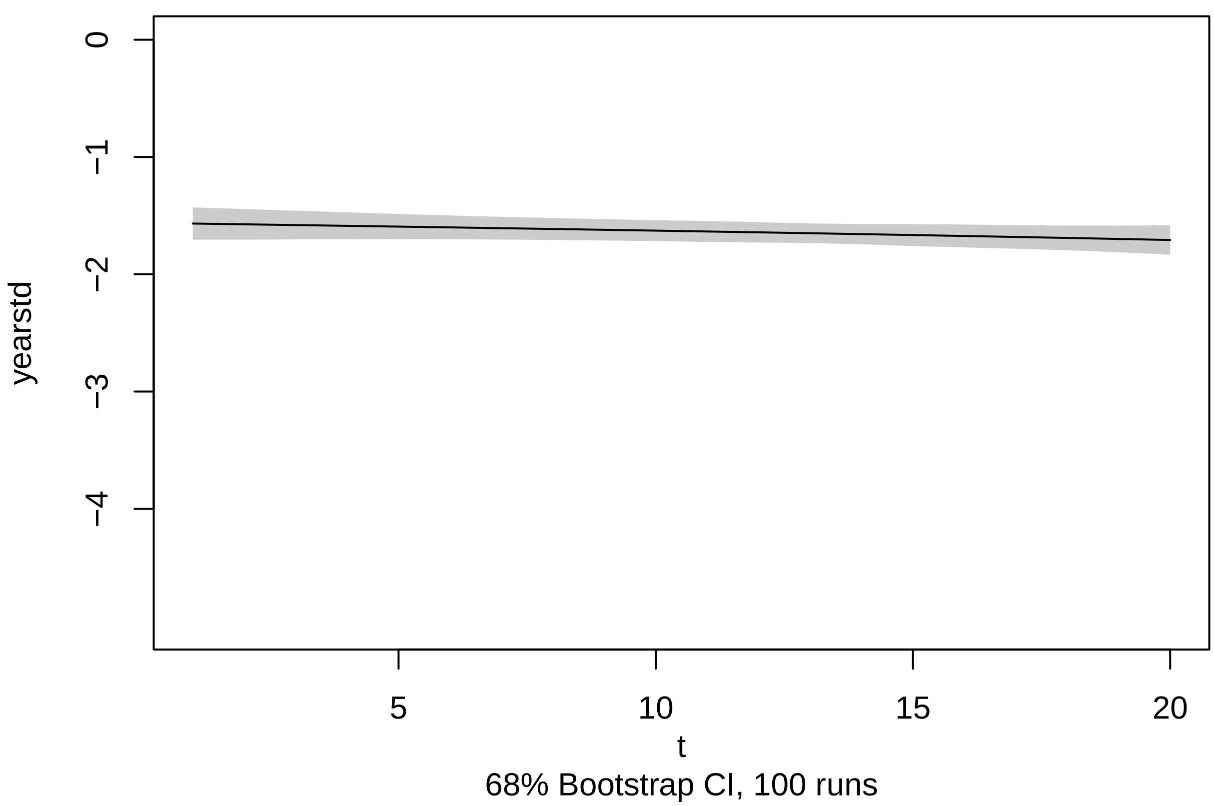
```
# see the time-varying pattern of the linear term's parameter:
plot(mod.tvfe.CI, vars = 1, ylim = c(1.5, 2.5))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), ylim = c(1.5, 2.5), ylab ="", xlab ="", type =
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[1, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[1
lines(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```
# see the time-varying pattern of the quadratic term's parameter:
plot(mod.tvfe.CI, vars = 2, ylim = c(-0.6, -0.2))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), ylim =c(-0.6, -0.2), ylab ="", xlab ="", type =
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[2, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[2
lines(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```r
# see the estimated trend function:
plot(mod.tvfe.CI, vars = 5, ylim = c(-5, 0))
```



```
graphics::par(mfrow = c(1, 5),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
```

2) in-sample forecasting MSE

```
fittedtvfe <- mod.tvfe$fitted
MSE_fe <- mean((mod.fe$residuals)^2)
MSE_fe
```

```
## [1] 0.0007329573
```

```
MSE_tvfe <- mean((mod.tvfe$residuals)^2)
MSE_tvfe
```

```
## [1] 0.00228338
```

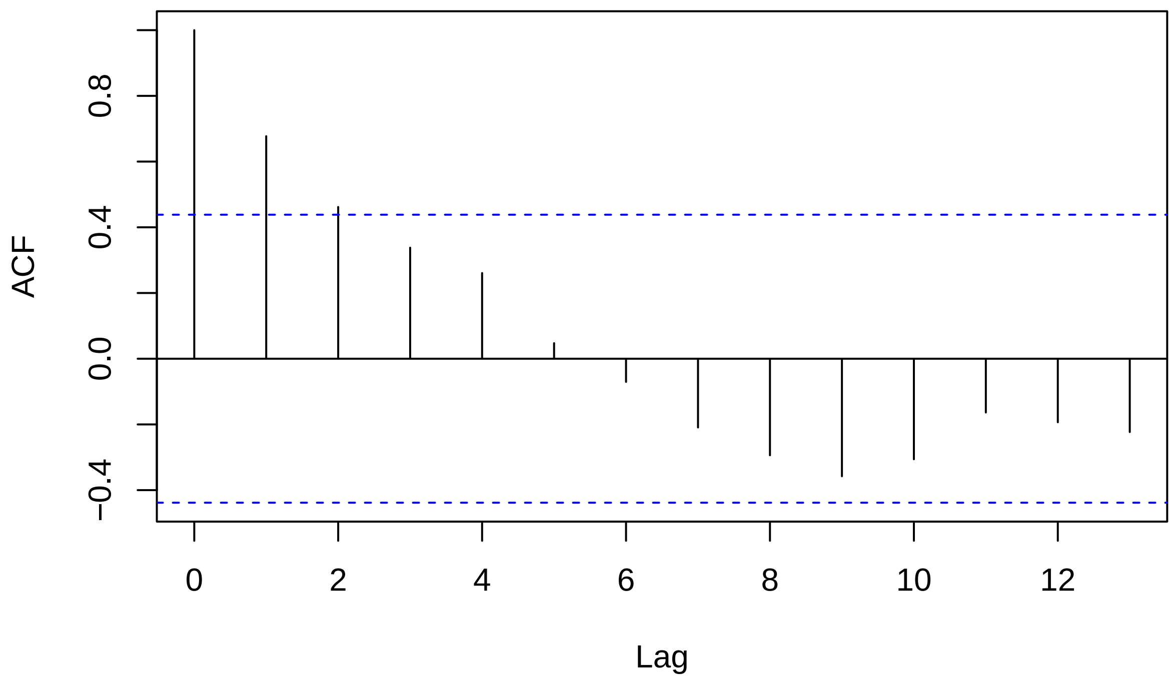
3) Residual's ACF

```
residtvfe <- mod.tvfe$residuals
residtvfe
```

```
##  [1]  0.0115350687  0.0423870009  0.0636118846  0.0344204403  0.0267938081
##  [6]  0.0150321380  0.0104533836  0.0101744747  0.0292572545  0.0333062540
## [11]  0.0450719132  0.0004259674  0.0598066360  0.0977387699  0.1129089082
## [16]  0.0786065412  0.1033150035  0.0862396359  0.0709936770  0.0653575245
## [21] -0.0351624687 -0.0161116091  0.0111124743  0.0115624744  0.0055740265
## [26]  0.0012997501 -0.0003862680  0.0006187770 -0.0027596994  0.0008038279
## [31] -0.0087346027 -0.0146079669 -0.0101717517  0.0242152033  0.0518447884
## [36]  0.0377587244 -0.0274515328 -0.0327500712 -0.0336533095 -0.0358193800
## [41]  0.0052888945 -0.0116758529 -0.0021913595 -0.0299313484 -0.0409216472
## [46] -0.0428856484 -0.0486283769 -0.0593343194 -0.0319636692 -0.0367698582
## [51] -0.0427046033 -0.0696896199 -0.0679494038 -0.0712458675 -0.0407313318
## [56] -0.0417863868 -0.0571807679 -0.0759540059 -0.0869016065 -0.0697978758
```

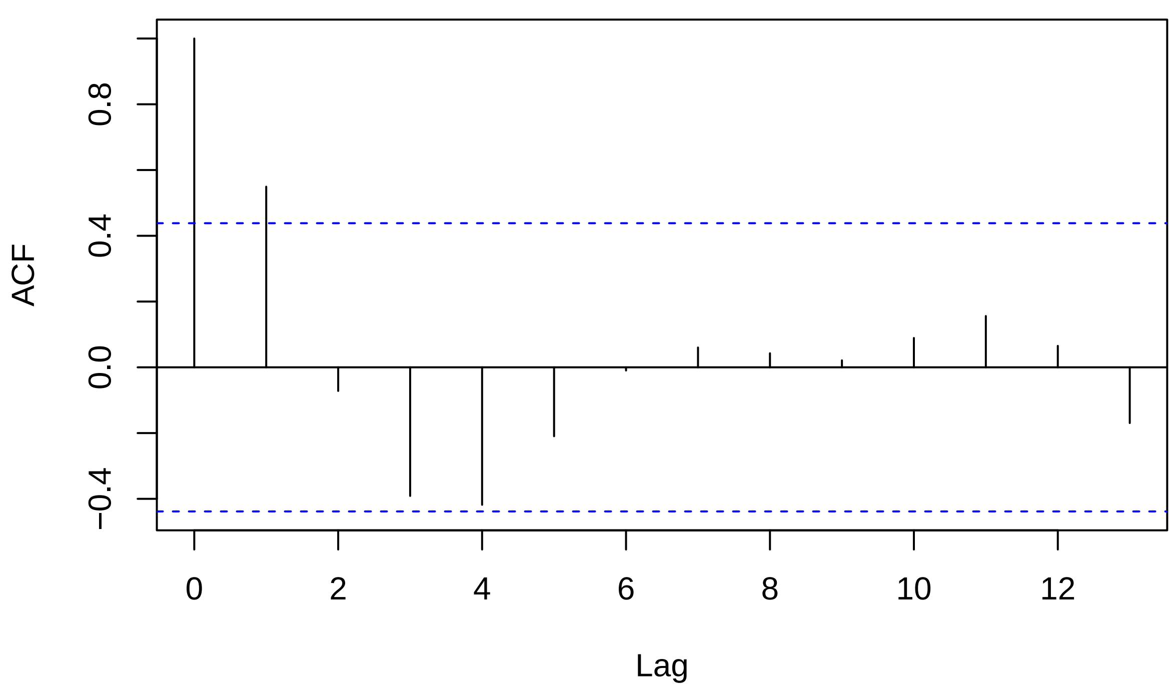
```
N=3
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```

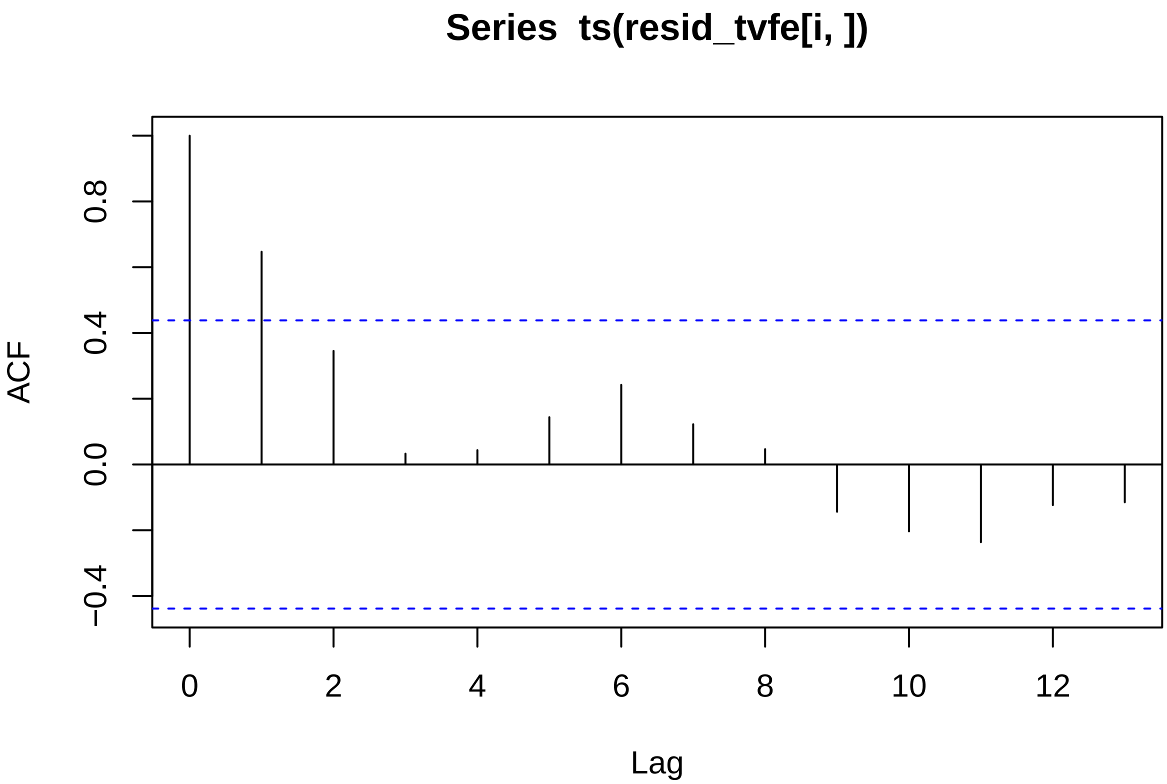
Series ts(resid_tvfe[i,])




```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value

## Warning in kpss.test(ts(data_i)): p-value greater than printed p-value
```

Series ts(resid_tvfe[i,])






```
pro_reject
```

```
##  [1]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

4) 3-steps out-of-sample predition MSE (to forecast the log(CO2) value in year 2017,2018 and 2019)

TVFE:

```
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
  forca <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
```

```r
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)

MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}

MSE
```

```
## [1] 0.002549891 0.003826993 0.003562487
```

FE:

```r
mod.tvfe <- tvReg::tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                 data = e, method ="pooling", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.25
```

```r
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
  forca <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)

MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}

MSE
```

```
## [1] 0.0009335055 0.0012527490 0.0012559288
```

#E (7) (K162)

```r
e_0915 <- read_excel("~/Desktop/RP/all_0915.xlsx", sheet = "E", range = "C1:K162")
# name the vectors:
colnames(e_0915) <- c("province","year","lco2","lgdp","lgdp2","ff","tech","region","yearstd")
e <- filter(e_0915, year<=2016)
ff<-e$ff
tech<-e$tech
```

FE

```r
mod.fe <- plm::plm(lco2~lgdp+lgdp2+ff+tech, index = c("province", "year"), model = "within", data=e)
mod.fe.CI <- confint(mod.fe, level = 0.68)
mod.fe.CI
```

```
##               16 %         84 %
## lgdp   1.613735993  1.762863728
## lgdp2 -0.310172364 -0.269548195
## ff    -0.002937538 -0.001223594
## tech   0.022609906  0.026308502
```

TVFE

Kernel: Gaussian Bandwidth: 0.6 Method: within

```r
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                  data = e, method ="within", bw =NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  1
```

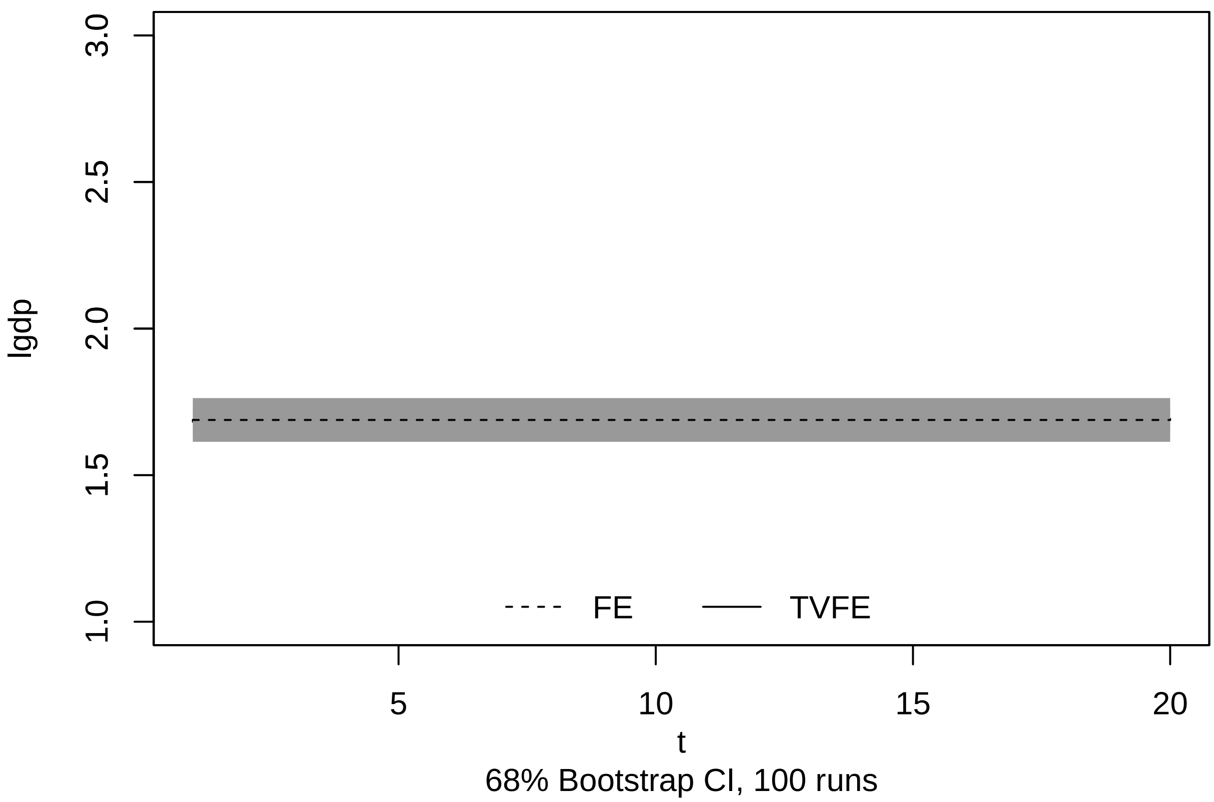
```r
# Bootstrapping to get 95% confidence intervals
mod.tvfe.CI <- confint(mod.tvfe, level = 0.68)
mod.tvfe.CI
```

```
##
## Class:  tvplm
##
## Mean of coefficient estimates:
## ===============================
##      lgdp     lgdp2        ff      tech   yearstd
##  1.687287 -0.289200 -0.002035  0.024499 -1.569223
##
## LOWER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  1.639516 -0.306719 -0.002861  0.022740 -1.626725
##
## UPPER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  1.735058 -0.271682 -0.001209  0.026259 -1.511720
##
## Bandwidth:  1
```

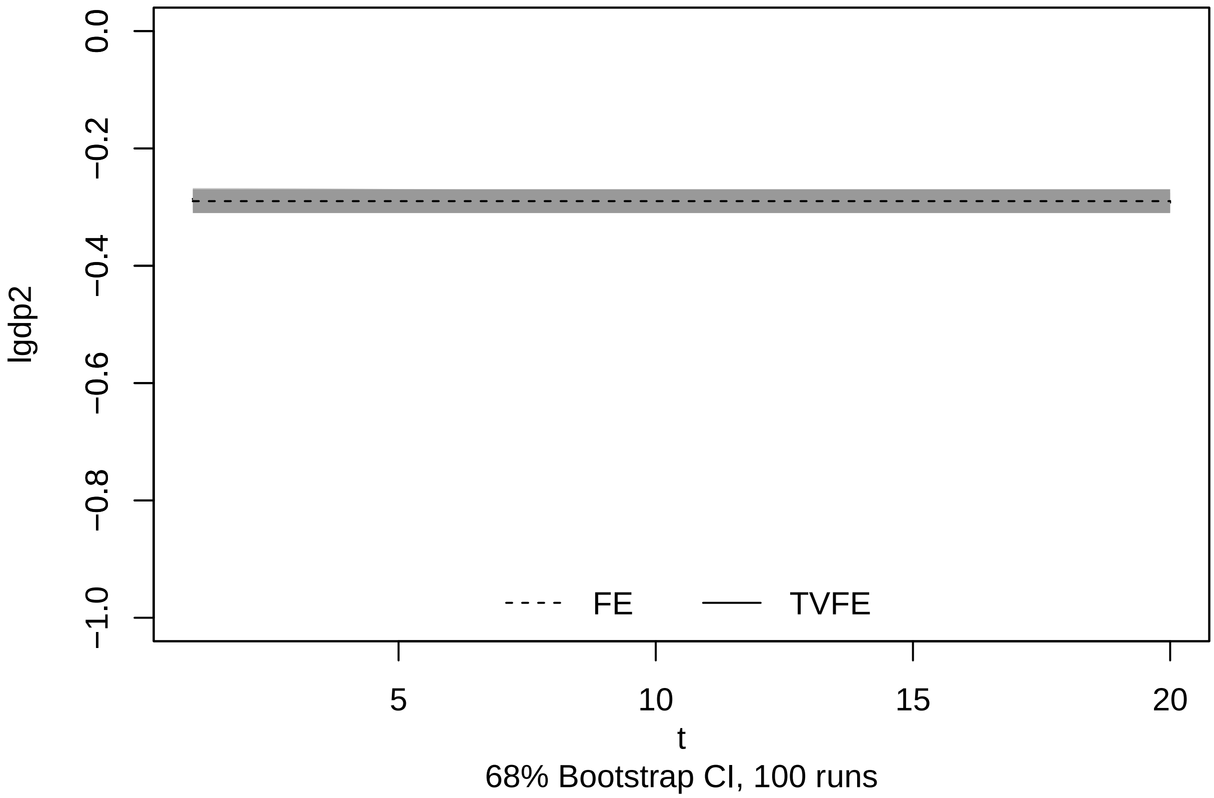
Post-estimation

1) Comparison plot of the within estimators

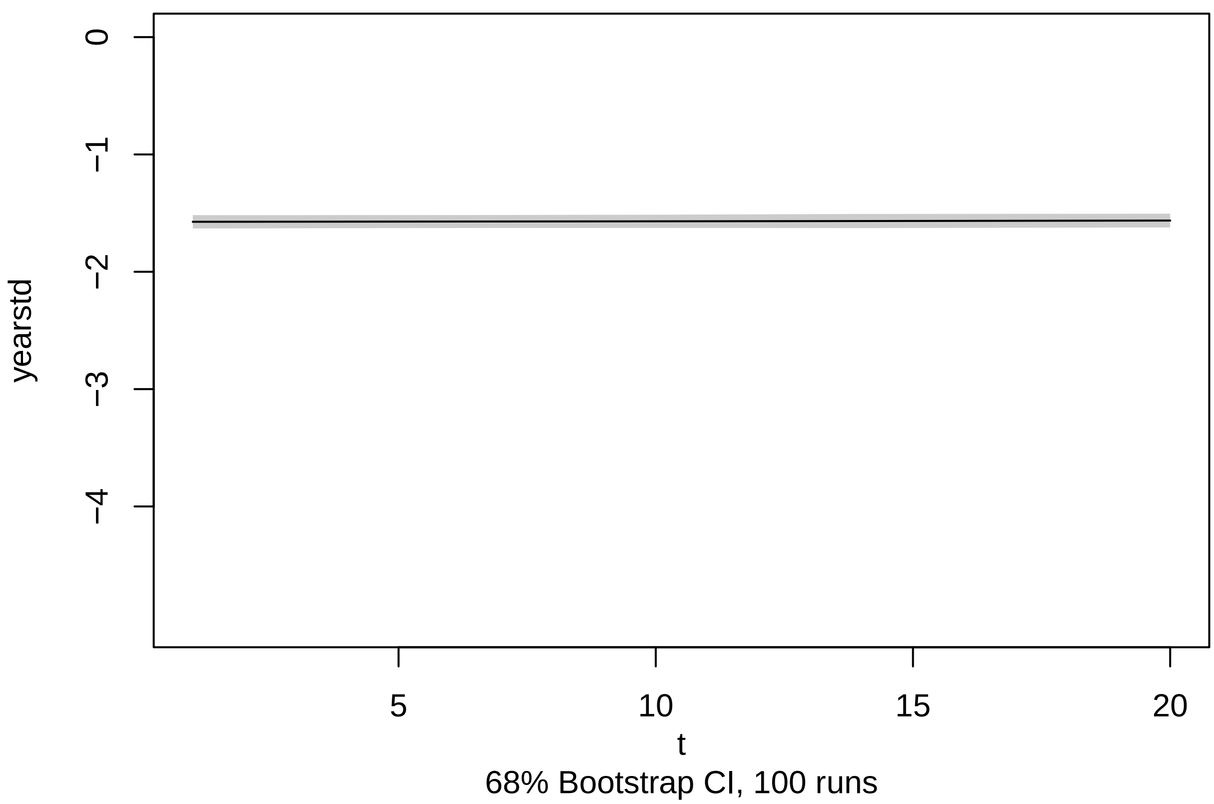
```
# see the time-varying pattern of the linear term's parameter:
plot(mod.tvfe.CI, vars = 1, ylim = c(1, 3))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), ylim = c(1, 3), ylab ="", xlab ="", type = "l",
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[1, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[1
lines(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```
# see the time-varying pattern of the quadratic term's parameter:
plot(mod.tvfe.CI, vars = 2, ylim = c(-1, 0))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), ylim = c(-1, 0), ylab ="", xlab ="", type = "l"
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[2, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[2
lines(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```
# see the estimated trend function:
plot(mod.tvfe.CI, vars = 5, ylim = c(-5, 0))
```



```
graphics::par(mfrow = c(1, 5),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
```

2) in-sample forecasting MSE

```
fittedtvfe <- mod.tvfe$fitted
MSE_fe <- mean((mod.fe$residuals)^2)
MSE_fe
```

```
## [1] 0.001461386
```

```
MSE_tvfe <- mean((mod.tvfe$residuals)^2)
MSE_tvfe
```

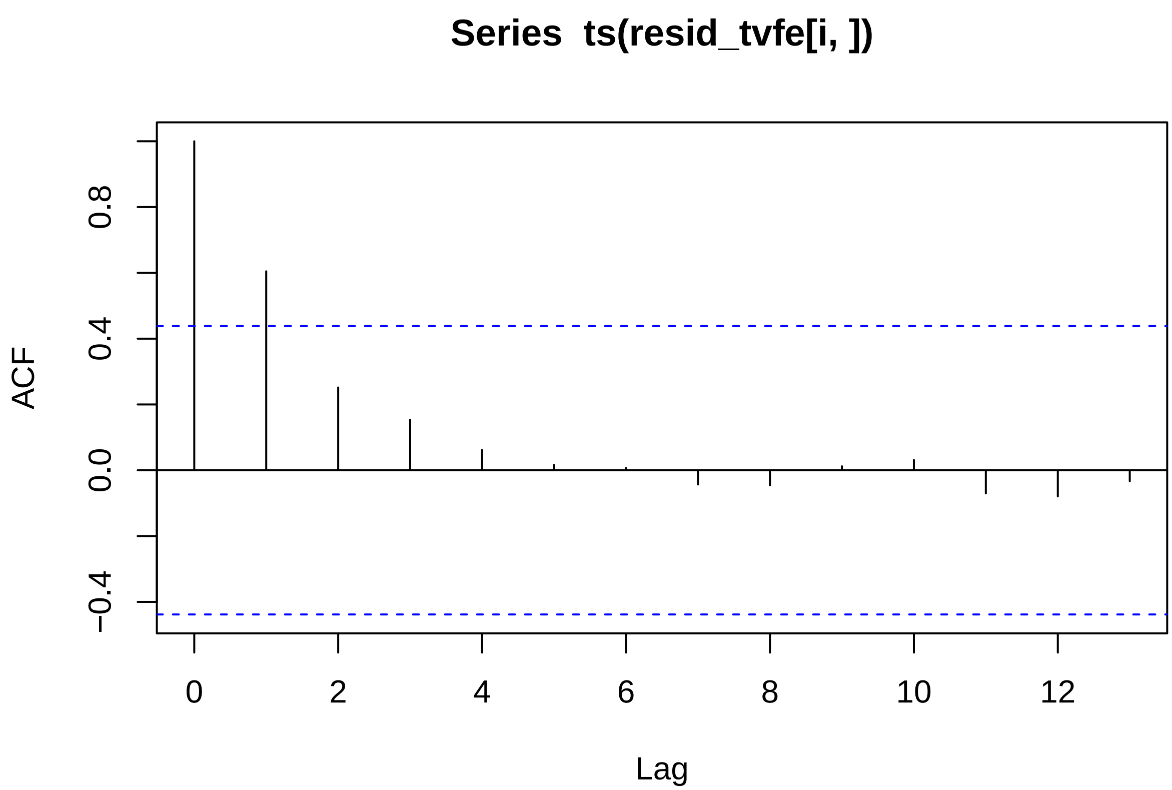
```
## [1] 0.004122096
```

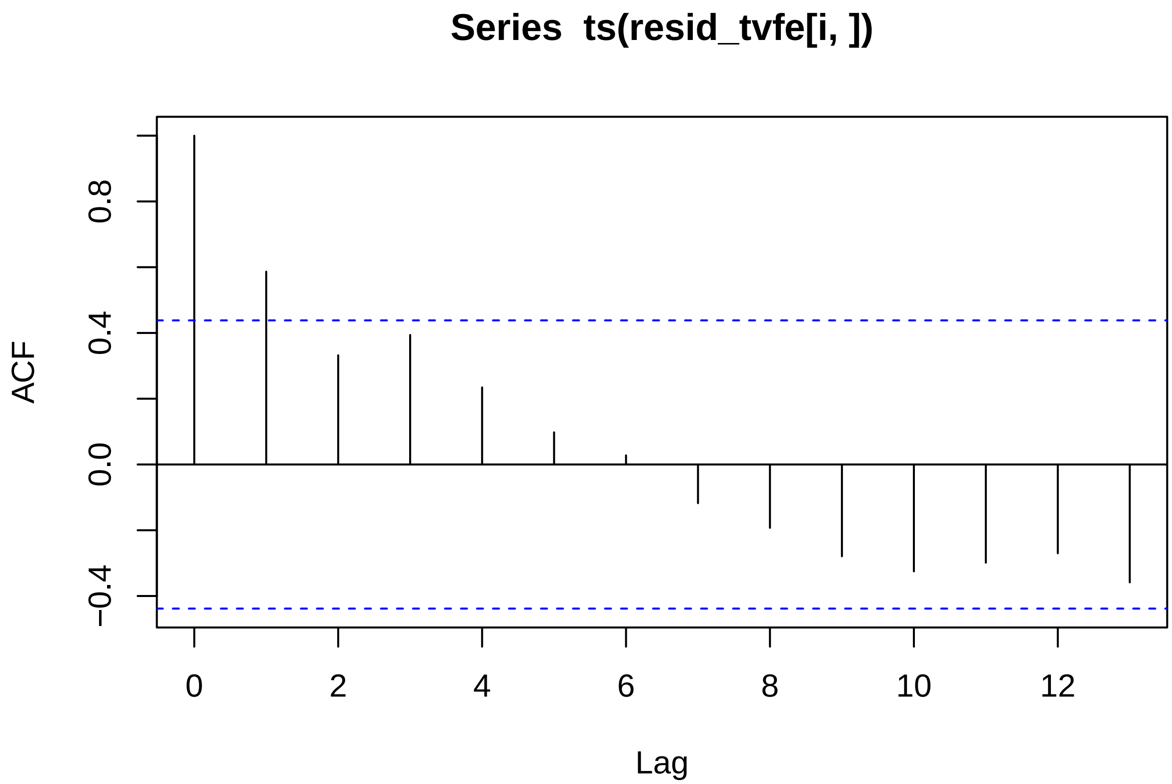
3) Residual's ACF

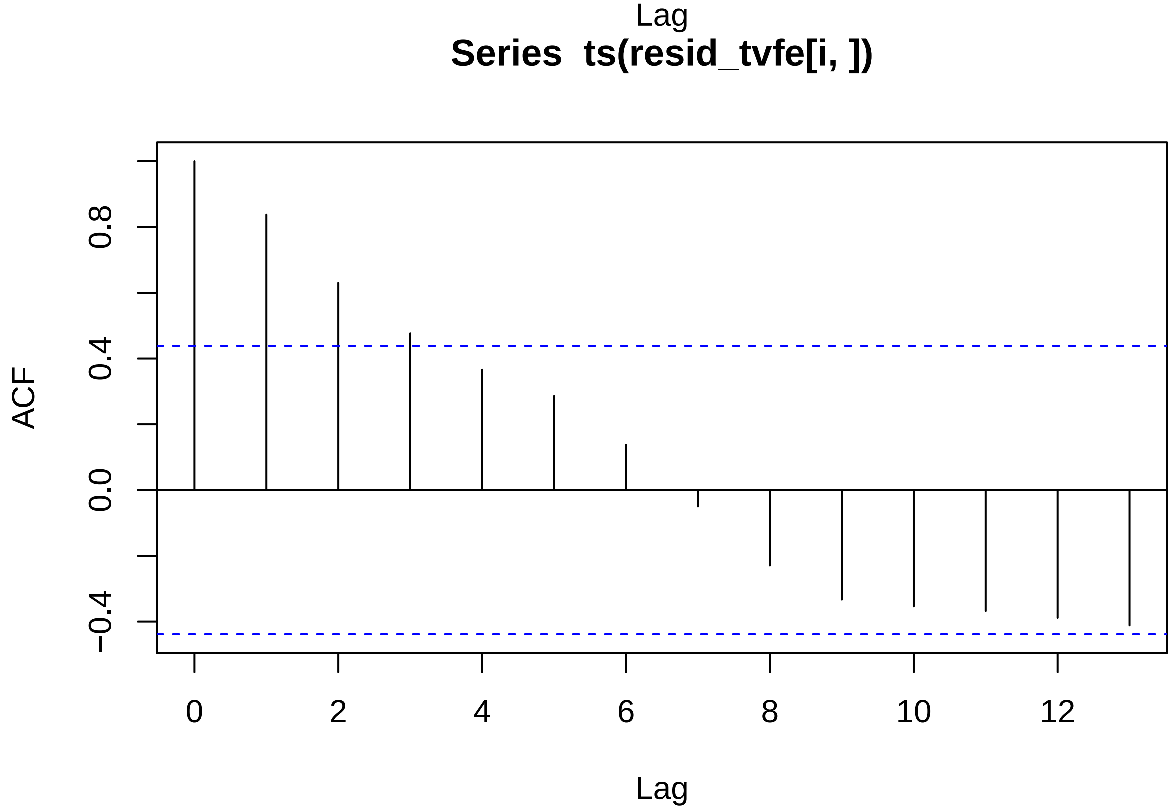
```
residtvfe <- mod.tvfe$residuals
residtvfe
```

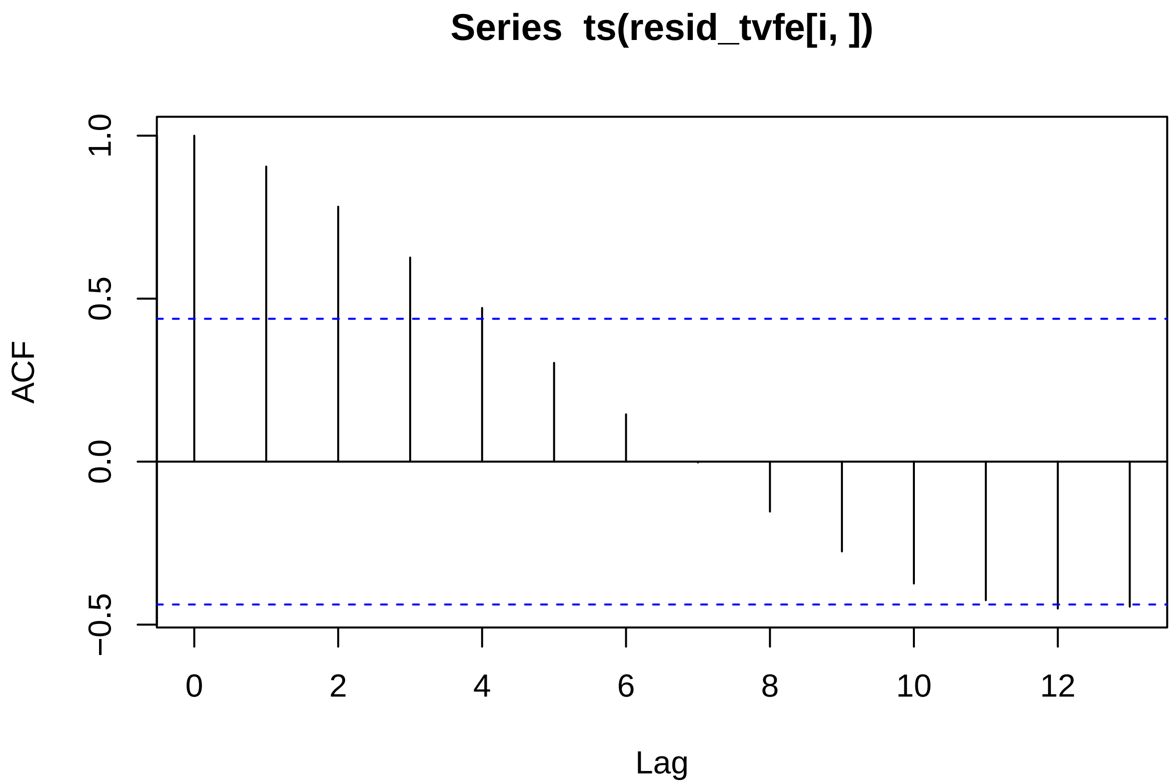
```
##   [1] -5.309464e-02 -2.773395e-02  2.109116e-04 -1.966973e-03 -7.675175e-04
##   [6] -2.719667e-05 -5.570332e-03 -3.648759e-03 -3.867879e-03 -5.926802e-03
##  [11] -1.170764e-02 -1.348959e-04  5.978533e-03  7.766181e-05  3.779736e-03
##  [16]  2.008183e-02  2.959445e-02  3.293243e-02  2.341257e-02  5.427479e-03
##  [21] -8.242351e-02 -1.024716e-01 -8.928653e-02 -9.323375e-02 -1.167386e-01
##  [26] -1.062276e-01 -8.827662e-02 -8.005496e-02 -7.581035e-02 -7.691642e-02
##  [31] -8.012522e-02 -8.526850e-02 -6.199049e-02 -5.932865e-02 -3.745807e-02
##  [36] -5.608699e-02 -8.219289e-02 -5.487436e-02 -5.403486e-02 -9.274809e-02
##  [41] -1.532656e-02 -1.299726e-02 -4.610873e-03 -6.476541e-03 -1.416027e-02
##  [46] -1.363745e-02 -8.902387e-03 -3.949612e-04  1.608478e-02  2.179908e-02
##  [51]  1.514055e-02  1.381896e-02  1.339271e-02  2.706404e-02  4.555569e-02
##  [56]  3.975488e-02  3.872615e-02  2.108689e-02  2.152287e-02  2.416487e-02
##  [61]  1.983687e-02  2.885556e-02  4.401188e-02  3.390425e-02  1.898106e-02
##  [66]  5.514860e-03 -1.908322e-04 -4.168555e-03 -1.626576e-02 -2.937111e-02
##  [71] -5.458352e-02 -6.986499e-02 -8.953372e-02 -9.454083e-02 -1.110455e-01
##  [76] -1.279671e-01 -1.186573e-01 -1.280238e-01 -1.284181e-01 -1.316756e-01
##  [81]  3.120571e-02  4.380503e-02  5.568496e-02  6.045851e-02  5.145273e-02
##  [86]  4.494902e-02  3.754159e-02  4.443054e-02  5.177883e-02  7.559906e-02
##  [91]  1.003653e-01  1.160313e-01  1.283803e-01  1.402571e-01  1.495507e-01
##  [96]  1.561807e-01  1.444748e-01  1.557955e-01  1.638279e-01  1.714301e-01
## [101]  3.563300e-02  3.595312e-02  7.695602e-02  8.001808e-02  7.500668e-02
## [106]  6.928568e-02  8.054824e-02  7.975561e-02  7.873914e-02  5.003863e-02
## [111]  4.725091e-03  1.313073e-02 -1.123061e-02  4.196570e-03  7.199293e-03
## [116] -7.630895e-03  6.155606e-03 -5.738672e-02 -4.713872e-02 -6.387125e-02
## [121] -1.279642e-02 -2.266054e-02 -1.901594e-02 -5.815974e-02 -1.975393e-02
## [126] -2.036710e-02 -1.973612e-02  4.164527e-03  1.592862e-02  2.598197e-02
## [131]  2.788319e-02  2.793999e-02  2.772091e-02  2.370739e-02  2.440189e-02
## [136]  7.086991e-04 -9.254024e-03 -2.593709e-02 -3.542465e-02 -4.910165e-02
```

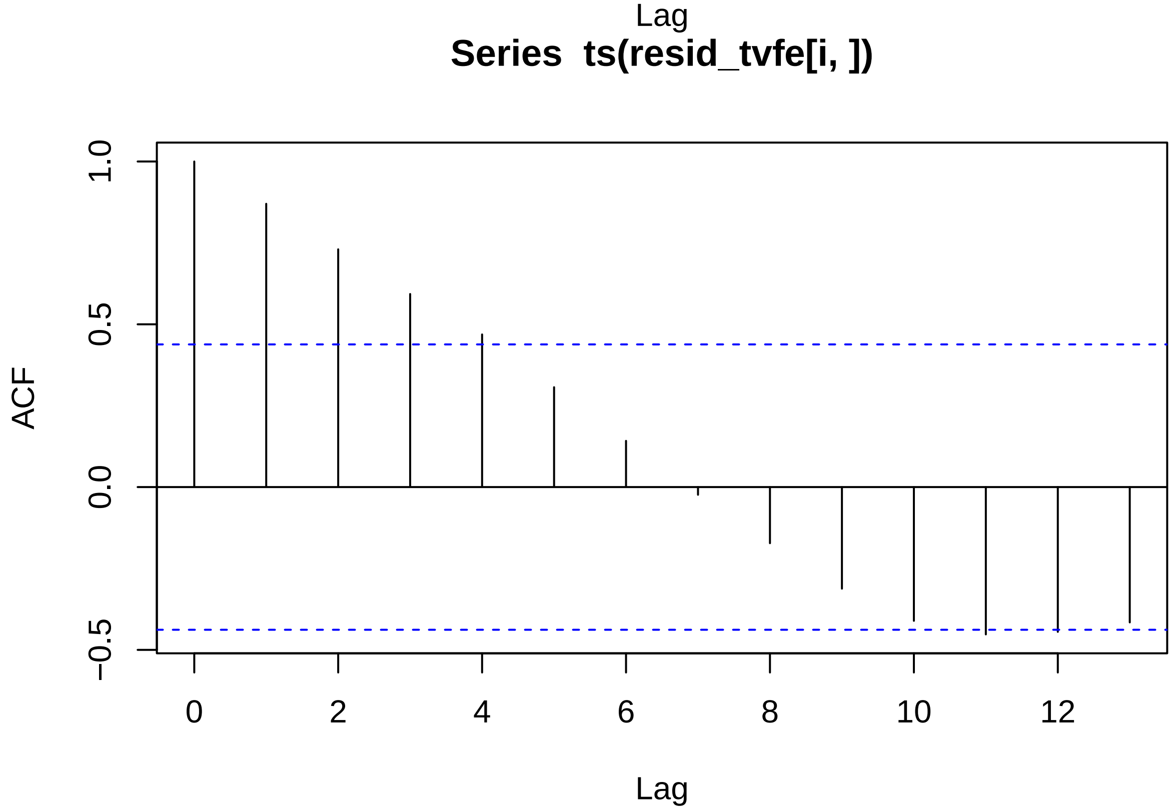
```
N=7
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```

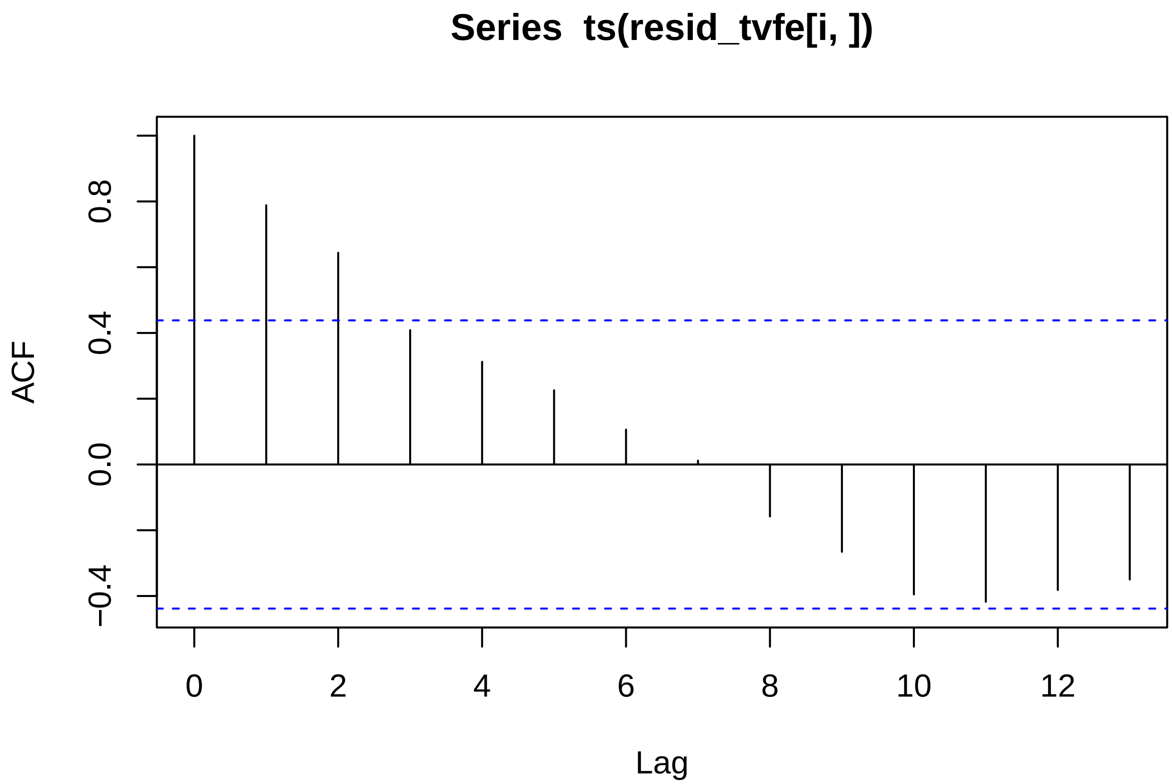



Series ts(resid_tvfe[i,])
ACF
Lag
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12

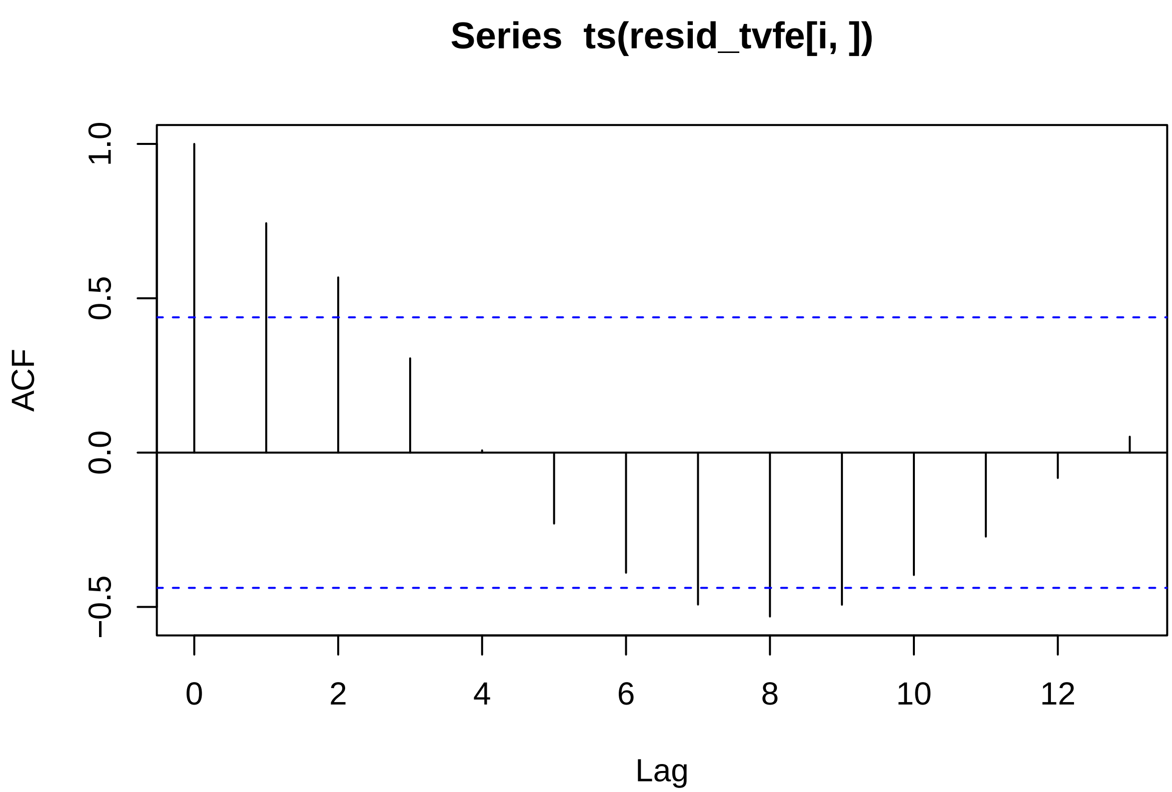

Series ts(resid_tvfe[i,])
ACF
Lag
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12


Series ts(resid_tvfe[i,])
ACF
1.0
0.5
0.0
−0.5
0
2
4
6
8
10
12
Lag


Series ts(resid_tvfe[i,])
ACF
1.0
0.5
0.0
−0.5
0
2
4
6
8
10
12
Lag




```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value

## Warning in kpss.test(ts(data_i)): p-value greater than printed p-value
```



```r
pro_reject
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

4) 3-steps out-of-sample predition MSE (to forecast the log(CO2) value in year 2017,2018 and 2019)

TVFE:

```r
mod.tvfe <- tvReg::tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                data = e, method ="pooling", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.25
```

```r
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
  forca <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)
```

```
## Warning in matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE): data length
## differs from size of matrix: [21 != 3 x 3]
```

```r
MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}
```

```
## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length
```

```
MSE
```

```
## [1] 0.007712446 0.009636972 0.007996831
```

FE:

```
library(plm)
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech")
  forca =c(0,0,0)
  forca <- predict(mod.fe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.
```

```
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)
```

```
## Warning in matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE): data length
## differs from size of matrix: [21 != 3 x 3]
```

```r
MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}
```

```
## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length
```

```r
MSE
```

```
## [1] 0.002857048 0.004070497 0.002868005
```

N (5)

```r
e_0915 <- read_excel("~/Desktop/RP/all_0915.xlsx", sheet = "N", range = "C1:K116")
# name the vectors:
colnames(e_0915) <- c("province","year","lco2","lgdp","lgdp2","ff","tech","region","yearstd")
e <- filter(e_0915, year<=2016)
ff<-e$ff
tech<-e$tech
```

FE

```r
mod.fe <- plm::plm(lco2~lgdp+lgdp2+ff+tech, index = c("province", "year"), model = "within", data=e)
summary(mod.fe)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm::plm(formula = lco2 ~ lgdp + lgdp2 + ff + tech, data = e,
##     model = "within", index = c("province", "year"))
##
## Balanced Panel: n = 5, T = 20, N = 100
##
## Residuals:
##       Min.    1st Qu.    Median   3rd Qu.      Max.
## -0.162918 -0.053000 -0.012166  0.055893  0.174948
```

```
##
## Coefficients:
##          Estimate  Std. Error  t-value  Pr(>|t|)
## lgdp   2.80451401  0.15642132  17.9292 < 2.2e-16 ***
## lgdp2 -0.64421996  0.04481342 -14.3756 < 2.2e-16 ***
## ff     0.00007468  0.00027373   0.2728    0.7856
## tech   0.01158613  0.00123193   9.4049 4.487e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    5.6922
## Residual Sum of Squares: 0.52489
## R-Squared:      0.90779
## Adj. R-Squared: 0.89968
## F-statistic: 223.964 on 4 and 91 DF, p-value: < 2.22e-16
```

```
mod.fe.CI <- confint(mod.fe, level=0.68)
mod.fe.CI
```

```
##              16 %          84 %
## lgdp   2.648959596  2.9600684297
## lgdp2 -0.688785012 -0.5996549032
## ff    -0.000197535  0.0003468949
## tech   0.010361029  0.0128112340
```

TVFE

Kernel: Gaussian Bandwidth: 0.25 Method: Pooling

```
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                 data = e, method ="within", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.3071443
```

```
summary(mod.tvfe)
```

```
##
## Call:
## tvPLM(formula = lco2 ~ lgdp + lgdp2 + ff + tech + yearstd, data = e,
##     index = c("province", "year"), bw = NULL, method = "within",
##     tkernel = "Gaussian")
##
## Class:  tvplm
##
## Summary of time-varying estimated coefficients:
## ================================================
##          lgdp   lgdp2         ff     tech yearstd
## Min.    2.339 -0.6824 -2.090e-04 0.009776  -2.495
## 1st Qu. 2.485 -0.6345 -1.855e-04 0.010244  -2.290
## Median  2.675 -0.6025 -6.661e-05 0.011811  -2.068
## Mean    2.673 -0.6036 -1.213e-05 0.011623  -2.067
```

```
## 3rd Qu. 2.846 -0.5676  1.347e-04 0.012954  -1.830
## Max.    3.051 -0.5413  3.450e-04 0.013179  -1.679
##
## Bandwidth:  0.3071
## Pseudo R-squared:  0.8002
```

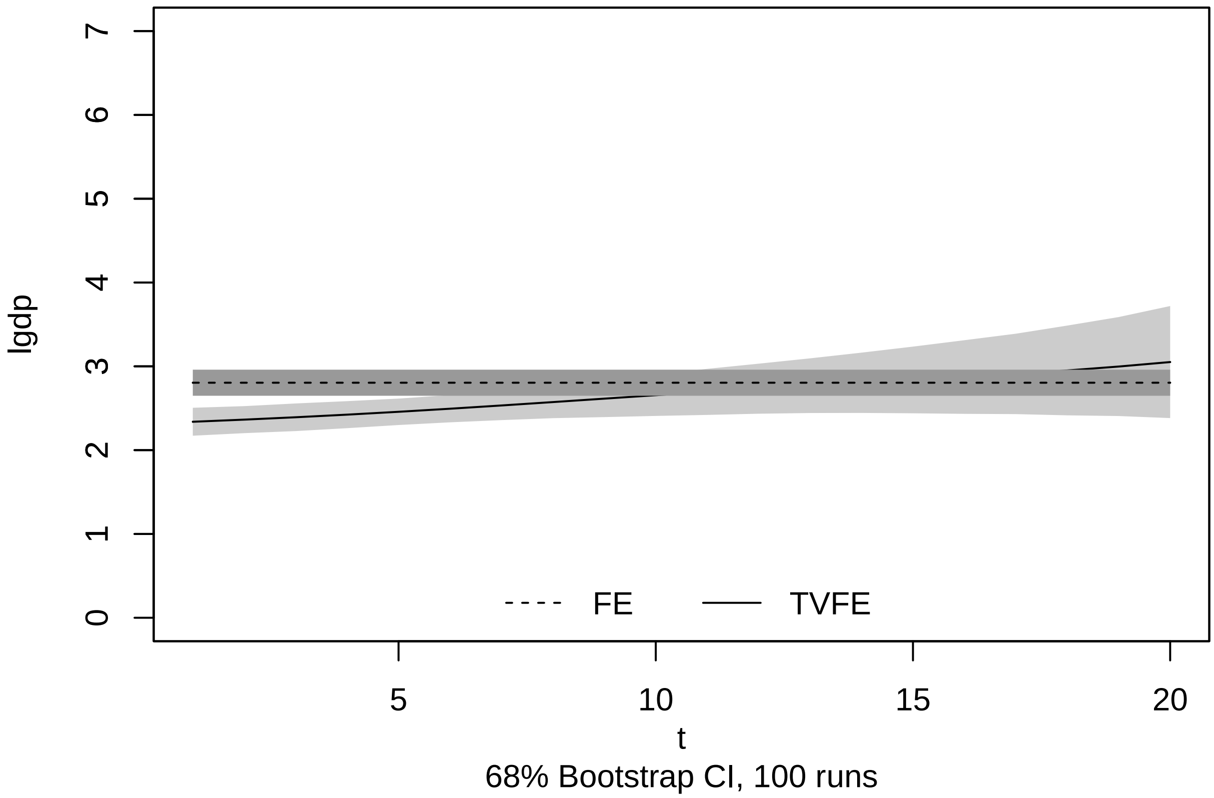
```
# Bootstrapping to get 95% confidence intervals
mod.tvfe.CI <- confint(mod.tvfe, level=0.68)
mod.tvfe.CI
```

```
##
## Class:  tvplm
##
## Mean of coefficient estimates:
## ===============================
##       lgdp      lgdp2         ff       tech    yearstd
##  2.673e+00 -6.036e-01 -1.213e-05  1.162e-02 -2.067e+00
##
## LOWER (68%):
##       lgdp      lgdp2         ff       tech    yearstd
##  2.3644703 -0.6969298 -0.0003486  0.0103348 -2.3510982
##
## UPPER (68%):
##       lgdp      lgdp2         ff       tech    yearstd
##  2.9822707 -0.5101916  0.0003243  0.0129115 -1.7824516
##
## Bandwidth:  0.3071
```

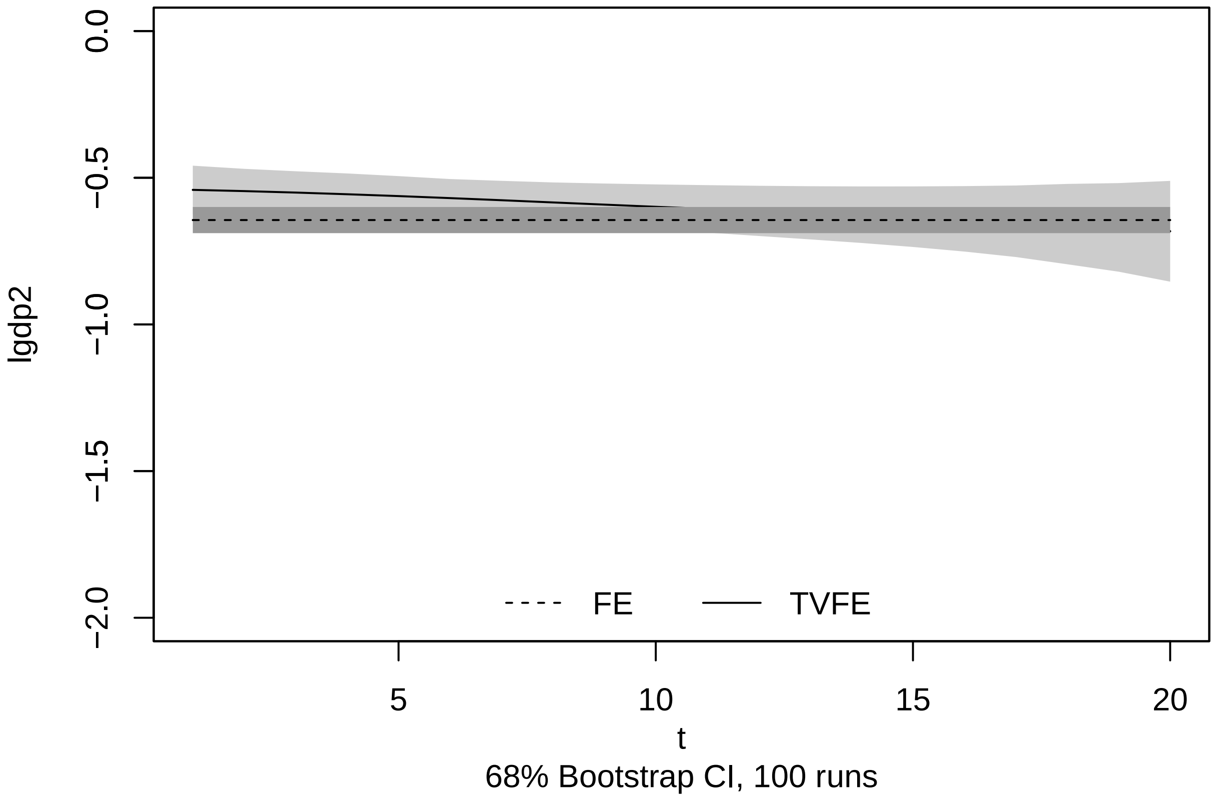
Post-estimation

1) plot of the within estimators

```
# see the time-varying pattern of the linear term's parameter:
plot(mod.tvfe.CI, vars = 1, ylim = c(0, 7))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), ylim = c(0, 7), ylab ="", xlab ="", type = "l",
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[1, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[1
lines(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```
# see the time-varying pattern of the quadratic term's parameter:
plot(mod.tvfe.CI, vars = 2, ylim = c(-2, 0))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), ylim = c(-2, 0), ylab ="", xlab ="", type = "l"
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[2, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[2
lines(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```



2) in-sample forecasting-Compare MSE

```
fittedtvfe <- mod.tvfe$fitted
MSE_fe <- mean((mod.fe$residuals)^2)
MSE_fe
```

```
## [1] 0.005248901
```

```
MSE_tvfe <- mean((mod.tvfe$residuals)^2)
MSE_tvfe
```

```
## [1] 0.01698182
```

3) Residual's ACF

```
residtvfe <- mod.tvfe$residuals
residtvfe
```

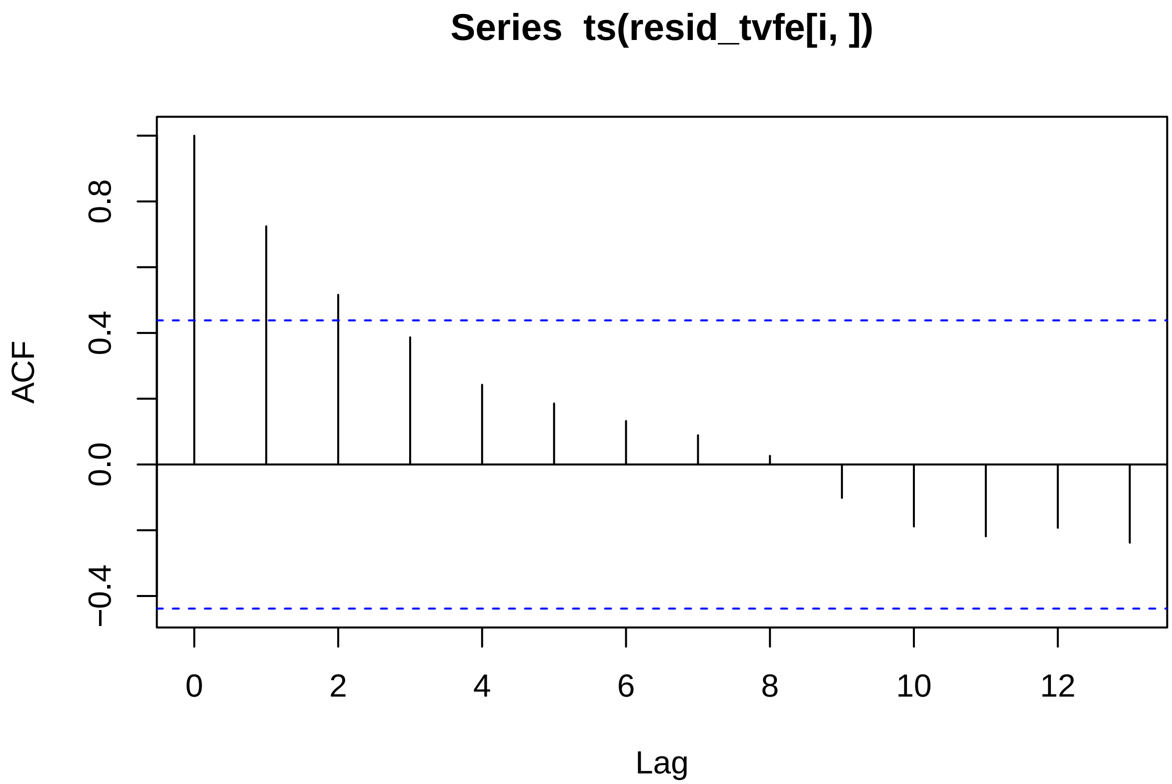
```
##   [1] -0.0470455811 -0.0815953041 -0.1144566712 -0.1373053202 -0.1807840352
##   [6] -0.2004169703 -0.2033457340 -0.1933373604 -0.1419029835 -0.2255893749
##  [11] -0.2558527369 -0.2252887032 -0.2313429100 -0.2461826620 -0.2646473544
##  [16] -0.2669533185 -0.3105732298 -0.3040481543 -0.3293093972 -0.3647684426
##  [21]  0.1122653752  0.1008631781  0.0881004516  0.0716667862  0.0570531819
```

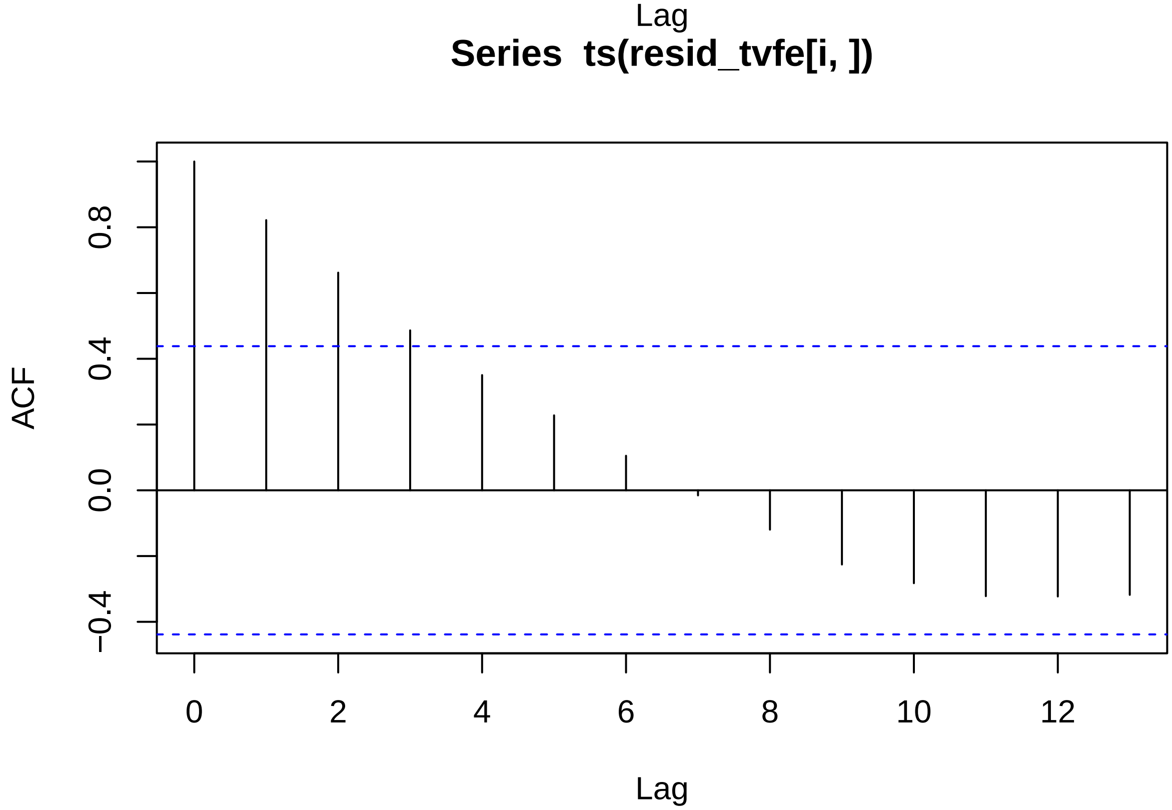
```
## [26]  0.0504716875  0.0359077328  0.0195325347  0.0114089998 -0.0133433806
## [31] -0.0226637616 -0.0272257026 -0.0274431461 -0.0245486439 -0.0161143993
## [36] -0.0149435542 -0.0151033748 -0.0358679318 -0.0339471441 -0.0541259629
## [41]  0.1056026053  0.1166107497  0.1005322252  0.0908737207  0.0777389454
## [46]  0.0666802668  0.0362005542  0.0419840892  0.0409852279  0.0462075631
## [51]  0.0616937256  0.0800500362  0.1054511544  0.1380599898  0.1708950056
## [56]  0.2022153410  0.2377762918  0.2602492793  0.2687113426  0.2850284869
## [61] -0.0669869942 -0.0247205446  0.0425187727  0.0299802275  0.0275806743
## [66]  0.0397596776 -0.0054178561  0.0211395891  0.0409751837  0.0456535301
## [71]  0.0018292325  0.0580561770  0.0629675330  0.0960284039  0.1316208472
## [76]  0.1481399252  0.0960747029  0.0984579409  0.1005516421  0.1200360862
## [81]  0.0117143089 -0.0007757512 -0.0104080003  0.0039341198 -0.0115931429
## [86] -0.0179813469 -0.0283198464 -0.0225218165 -0.0096996985 -0.0227339743
## [91] -0.0430147743 -0.0583825668 -0.0572066311  0.0257519015  0.0510373190
## [96]  0.0303653851  0.0446342957  0.0247567818 -0.0018714202 -0.0153129588
```

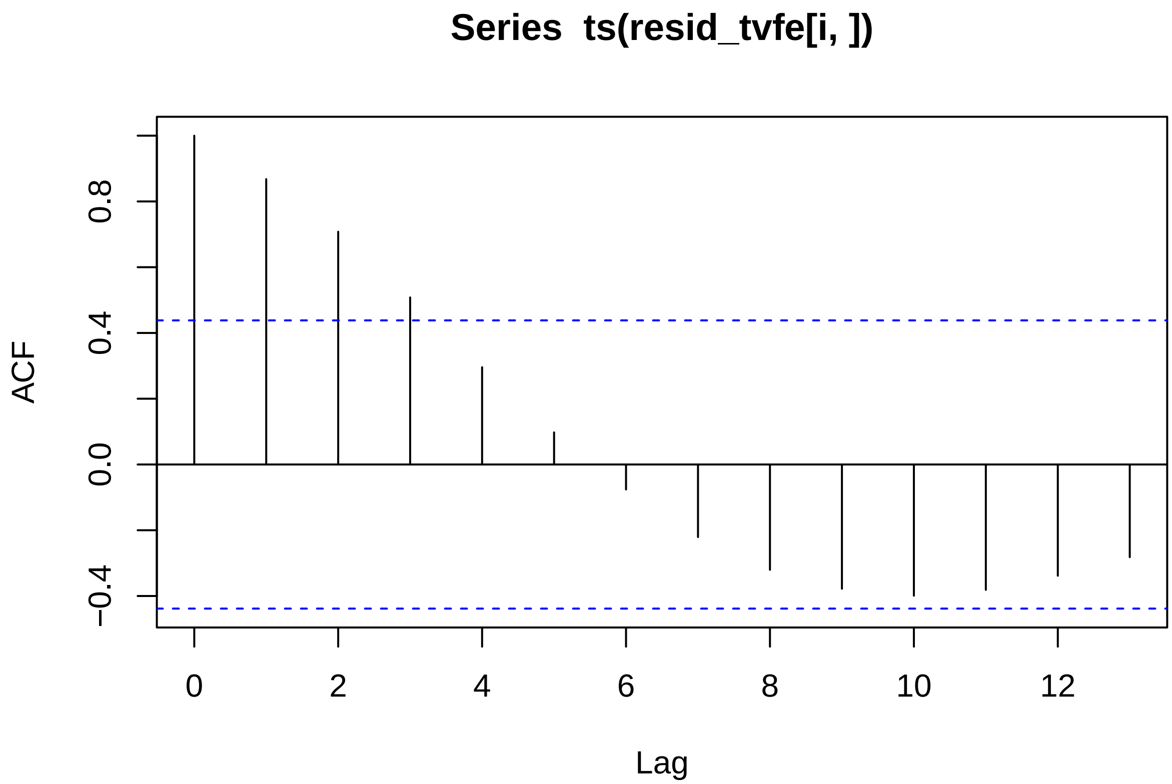
```
N=5
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```

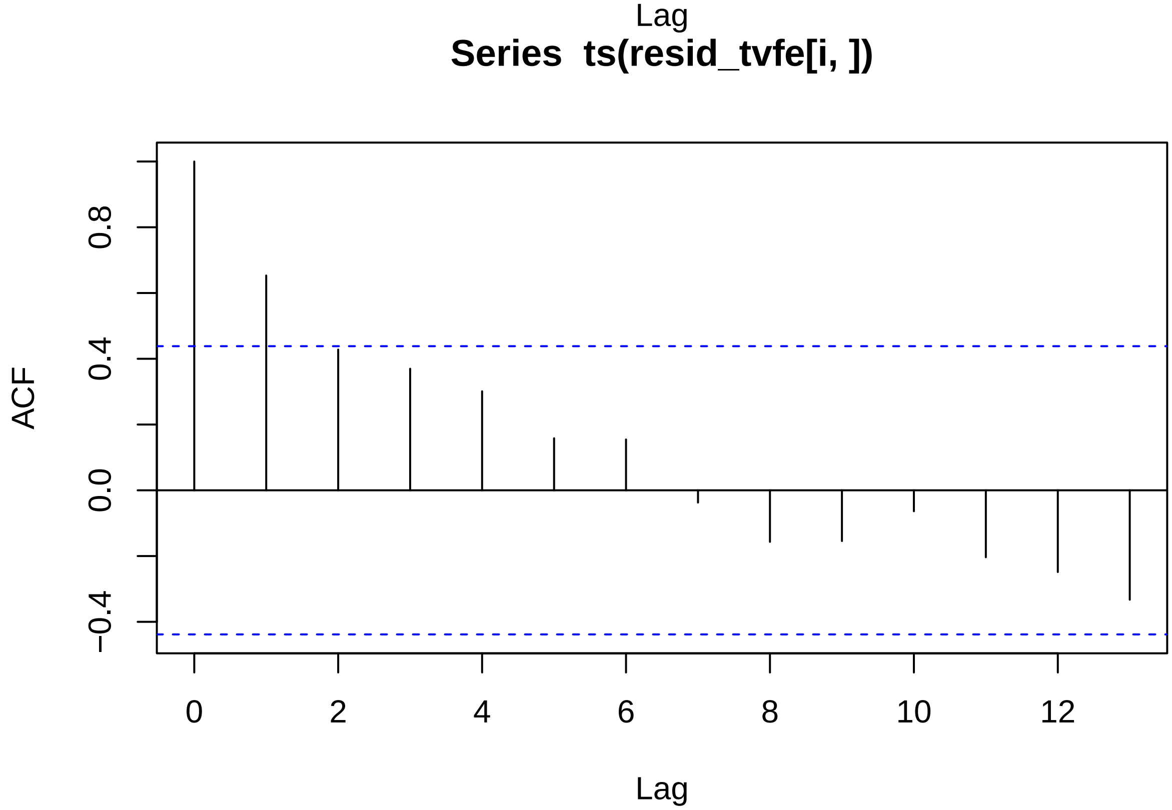
```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## smaller than printed p-value
```

```
## Warning in kpss.test(ts(data_i)): p-value smaller than printed p-value
```


Series ts(resid_tvfe[i,])
ACF
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12
Lag

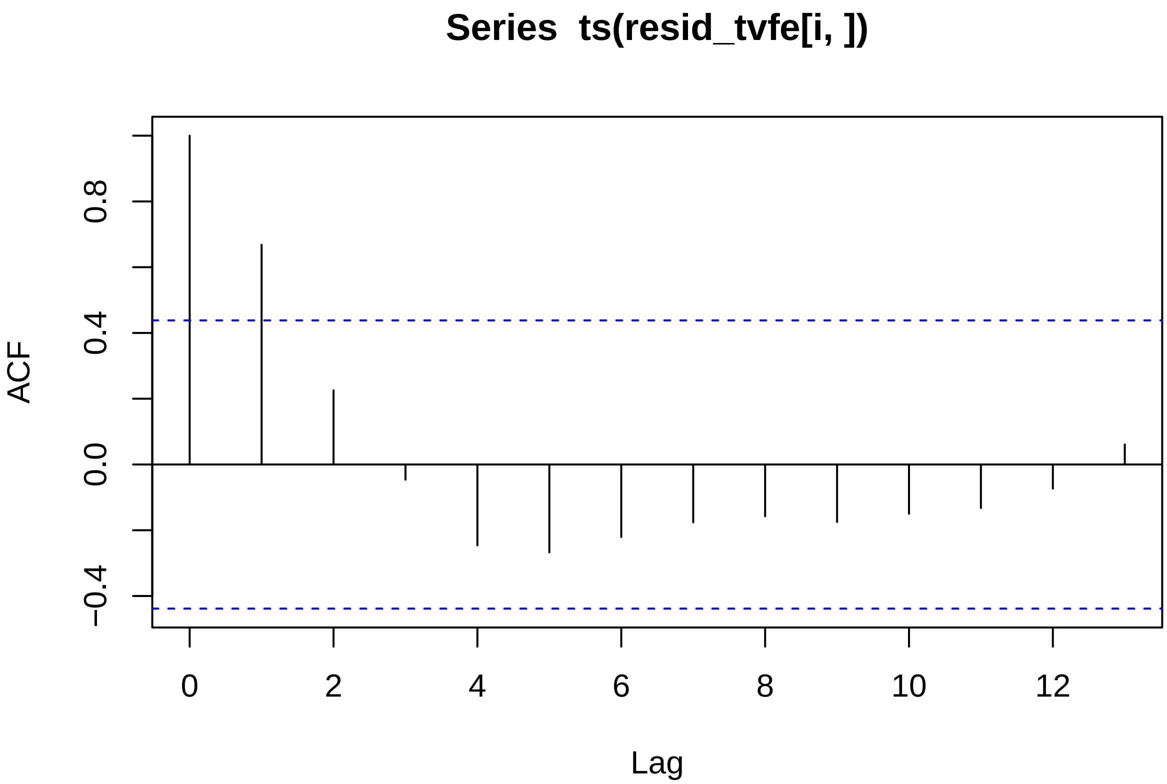

Series ts(resid_tvfe[i,])
ACF
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12
Lag






```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value
```

```
## Warning in kpss.test(ts(data_i)): p-value greater than printed p-value
```




```r
pro_reject
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

4) 3-steps out-of-sample predition MSE (to forecast the log(CO2) value in year 2017,2018 and 2019)

TVFE:

```r
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                  data = e, method ="pooling", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.25
```

```r
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
```

```r
  force <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)
```

```
## Warning in matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE): data length
## differs from size of matrix: [15 != 3 x 3]
```

```r
MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}
```

```
## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length
```

```r
MSE
```

```
## [1] 0.2603940 0.2614689 0.2571268
```

FE:

```r
library(plm)
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech")
  forca =c(0,0,0)
  forca <- predict(mod.fe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.
```

```
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)
```

```
## Warning in matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE): data length
## differs from size of matrix: [15 != 3 x 3]
```

```
MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}
```

```
## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length
```

```
MSE
```

```
## [1] 0.1943526 0.1914771 0.1885381
```

#NE(3)

```
e_0915 <- read_excel("~/Desktop/RP/all_0915.xlsx", sheet = "NE", range = "C1:K70")
# name the vectors:
colnames(e_0915) <- c("province","year","lco2","lgdp","lgdp2","ff","tech","region","yearstd")
e <- filter(e_0915, year<=2016)
ff<-e$ff
tech<-e$tech
```

FE

```r
mod.fe <- plm::plm(lco2~lgdp+lgdp2+ff+tech, index = c("province", "year"), model = "within", data=e)
mod.fe.CI <- confint(mod.fe, level = 0.68)
mod.fe.CI
```

```
##                16 %          84 %
## lgdp   1.4735609959  1.6882513821
## lgdp2 -0.3011193642 -0.2397921244
## ff    -0.0002336654 -0.0000463541
## tech   0.0109951871  0.0129081307
```

TVFE

Kernel: Gaussian Bandwidth: 0.6 Method: within

```r
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                  data = e, method ="within", bw =NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.25
```

```r
# Bootstrapping to get 95% confidence intervals
mod.tvfe.CI <- confint(mod.tvfe, level = 0.68)
mod.tvfe.CI
```

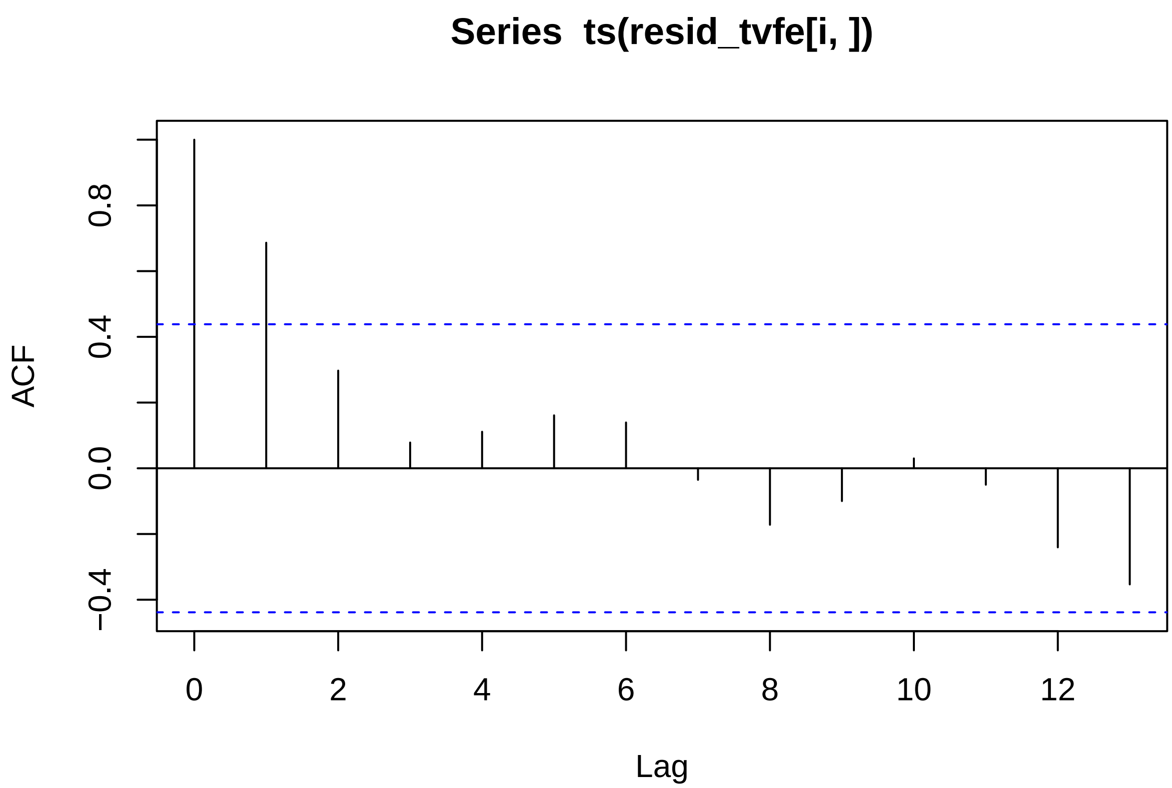
```
##
## Class:  tvplm
##
## Mean of coefficient estimates:
## ===============================
##       lgdp      lgdp2         ff       tech    yearstd
##  1.6638151 -0.2711521 -0.0001268  0.0143932 -1.3473827
##
## LOWER (68%):
##       lgdp      lgdp2         ff       tech    yearstd
##  1.4402600 -0.3280026 -0.0005506  0.0107075 -1.6365337
##
## UPPER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  1.887370 -0.214302  0.000297  0.018079 -1.058232
##
## Bandwidth:  0.25
```

```r
residtvfe <- mod.tvfe$residuals
residtvfe
```

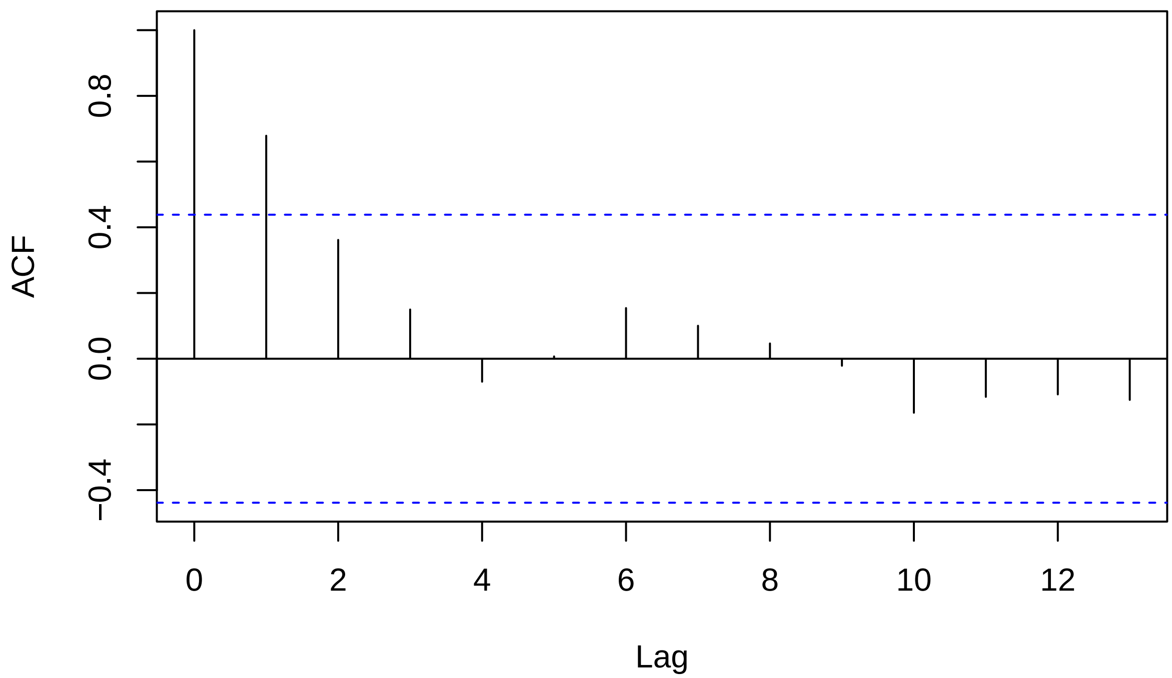
```
##  [1] -0.0299086554 -0.0203957179 -0.0170282207 -0.0112842136 -0.0133602772
##  [6] -0.0132586930 -0.0131839386 -0.0086463223  0.0007326415 -0.0058545071
## [11] -0.0248844612 -0.0251151748 -0.0053563374  0.0172846576  0.0215196227
## [16]  0.0228276008  0.0069545267  0.0011758841  0.0061361314  0.0151714125
```

```
## [21]  0.0045219100  0.0080891285  0.0041243543  0.0009392622 -0.0010499586
## [26] -0.0054198398 -0.0025705788 -0.0077309952 -0.0033600487 -0.0038856282
## [31] -0.0228624526 -0.0122889704 -0.0160992703 -0.0132198268  0.0023445916
## [36] -0.0029286728 -0.0326472306 -0.0373523738 -0.0499753197 -0.0542166605
## [41]  0.0110555089  0.0142402626  0.0125747549  0.0178562644  0.0138250115
## [46]  0.0164165360  0.0192434470  0.0233899572  0.0284138120  0.0305436306
## [51]  0.0211416829  0.0220679435  0.0228928581  0.0292813989  0.0323452699
## [56]  0.0350254376  0.0268632764  0.0219465352  0.0158088280  0.0193215284
```

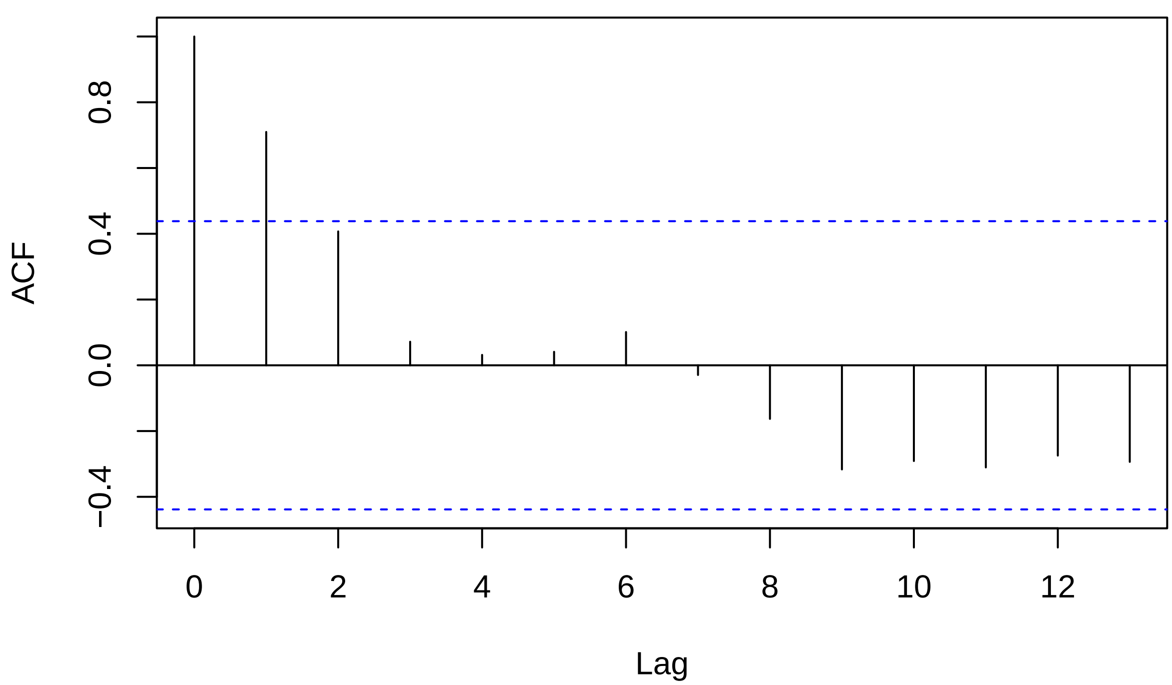
```r
N=3
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```



Series ts(resid_tvfe[i,])



Series ts(resid_tvfe[i,])




```
pro_reject
```

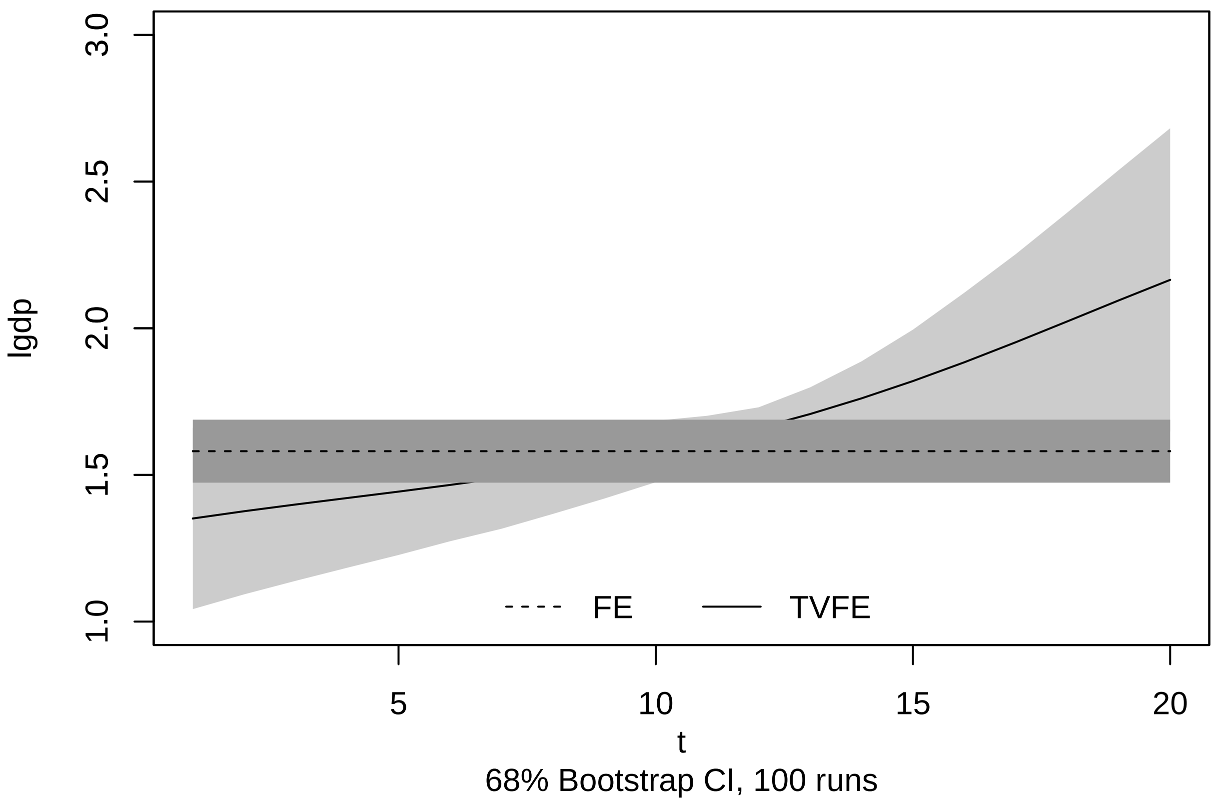
```
##  [1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

Post-estimation

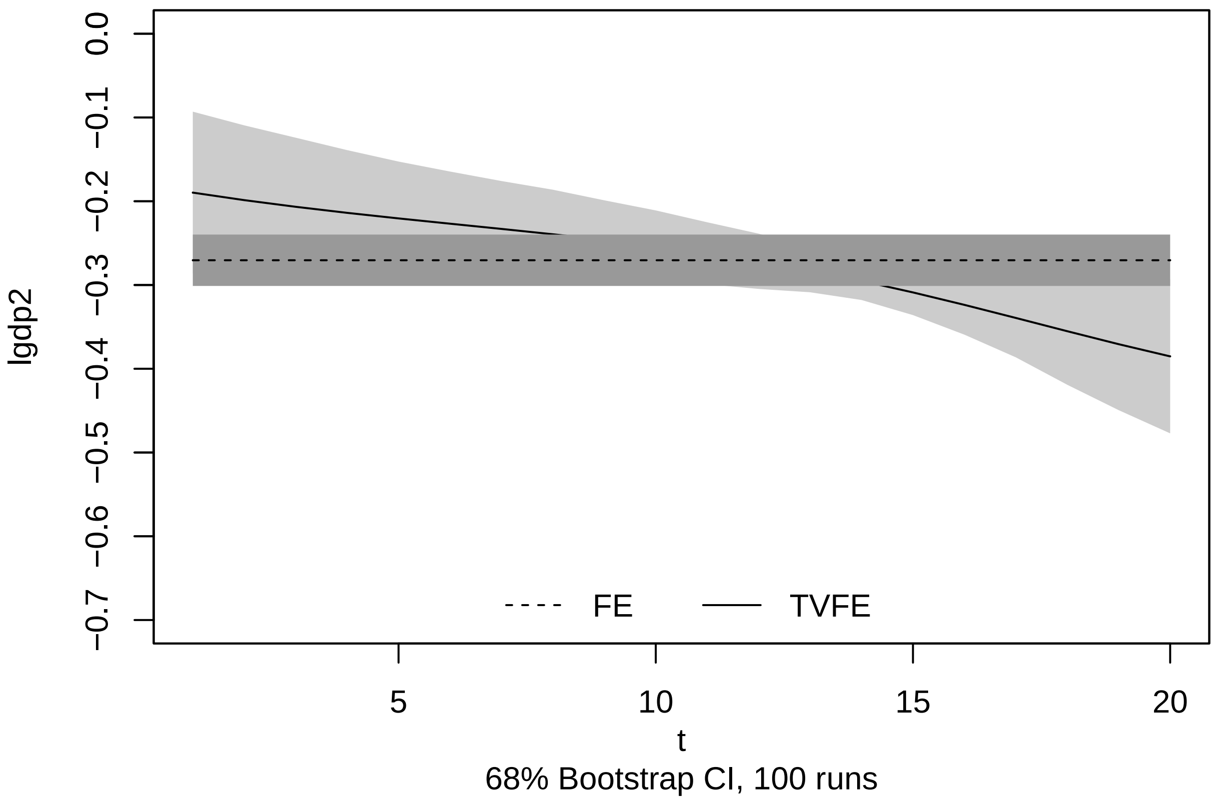
1) Comparison plot of the within estimators

```
# see the time-varying pattern of the linear term's parameter:
plot(mod.tvfe.CI, vars = 1, ylim = c(1, 3))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), ylim = c(1, 3), ylab ="", xlab ="", type = "l",
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[1, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[1
lines(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```

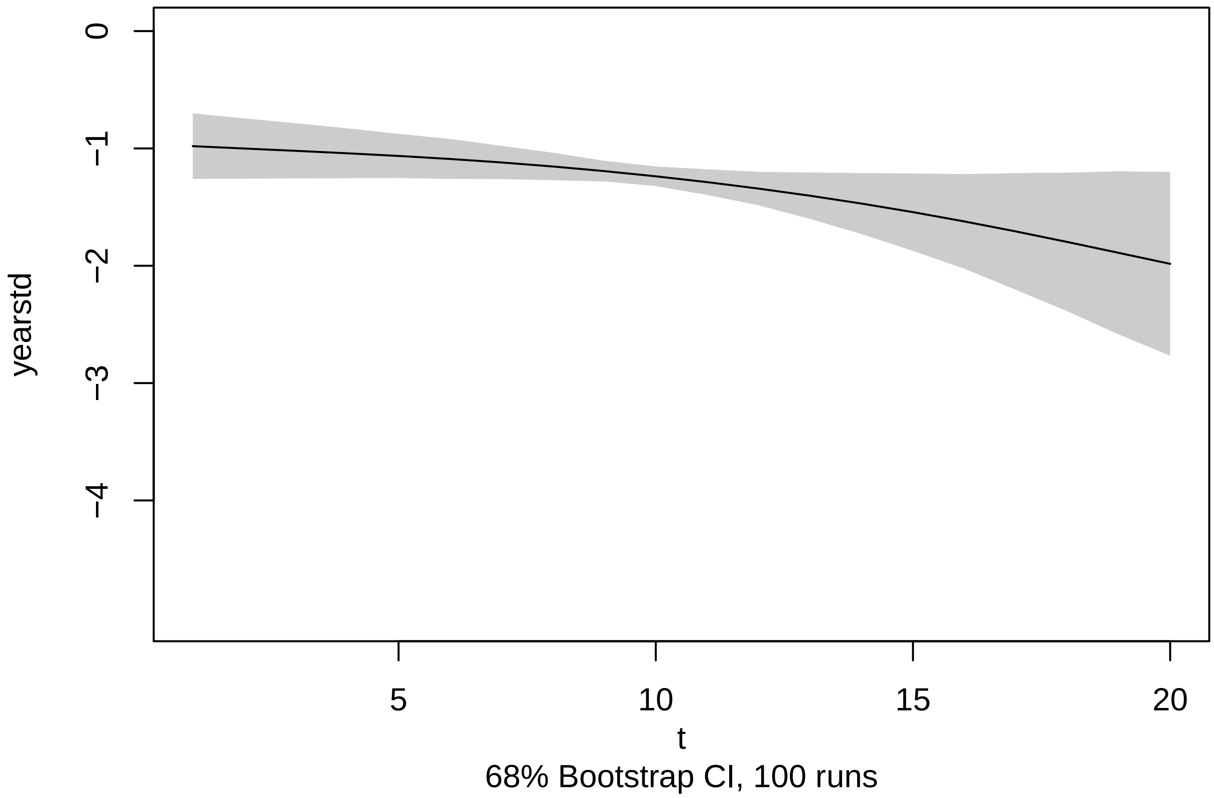



```
# see the time-varying pattern of the quadratic term's parameter:
plot(mod.tvfe.CI, vars = 2, ylim = c(-0.7, 0))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), ylim = c(-0.7, 0), ylab ="", xlab ="", type = "l
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[2, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[2
```

```r
lines(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```r
# see the estimated trend function:
plot(mod.tvfe.CI, vars = 5, ylim = c(-5, 0))
```




```
graphics::par(mfrow = c(1, 5),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
```

2) in-sample forecasting MSE

```
fittedtvfe <- mod.tvfe$fitted
MSE_fe <- mean((mod.fe$residuals)^2)
MSE_fe
```

```
## [1] 0.0003531462
```

```
MSE_tvfe <- mean((mod.tvfe$residuals)^2)
MSE_tvfe
```

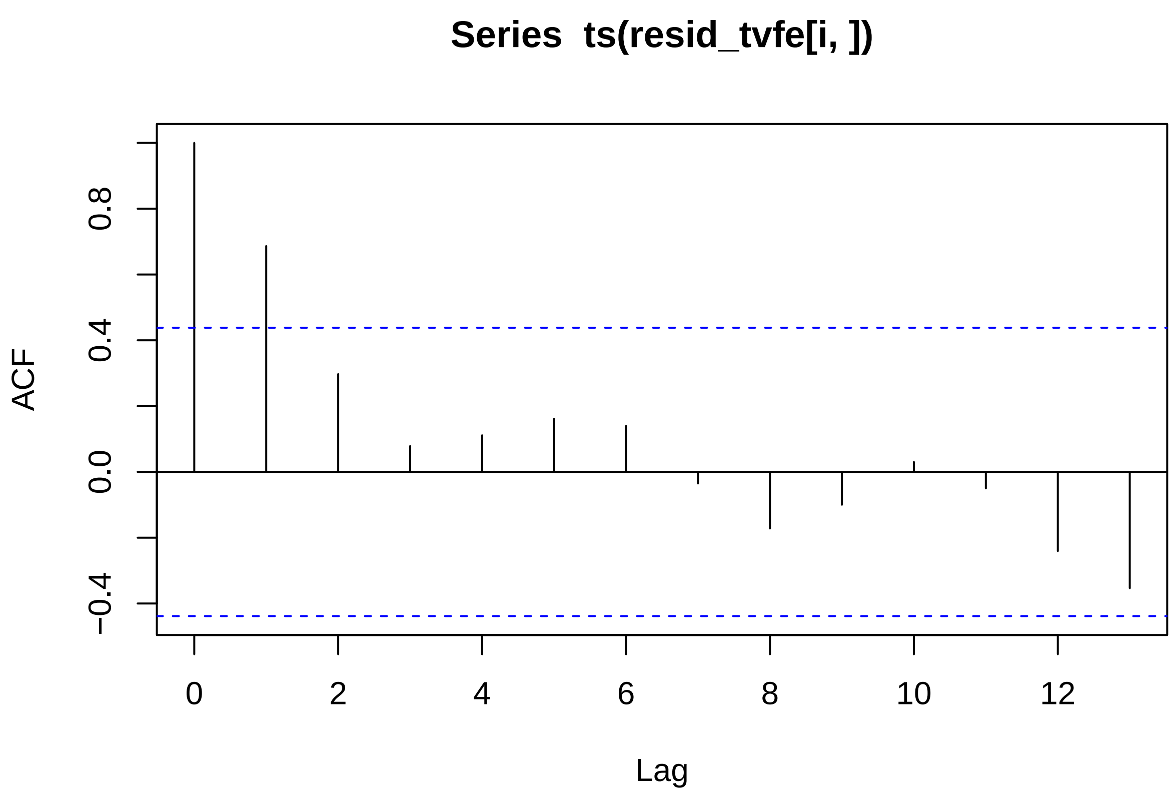
```
## [1] 0.0004137787
```

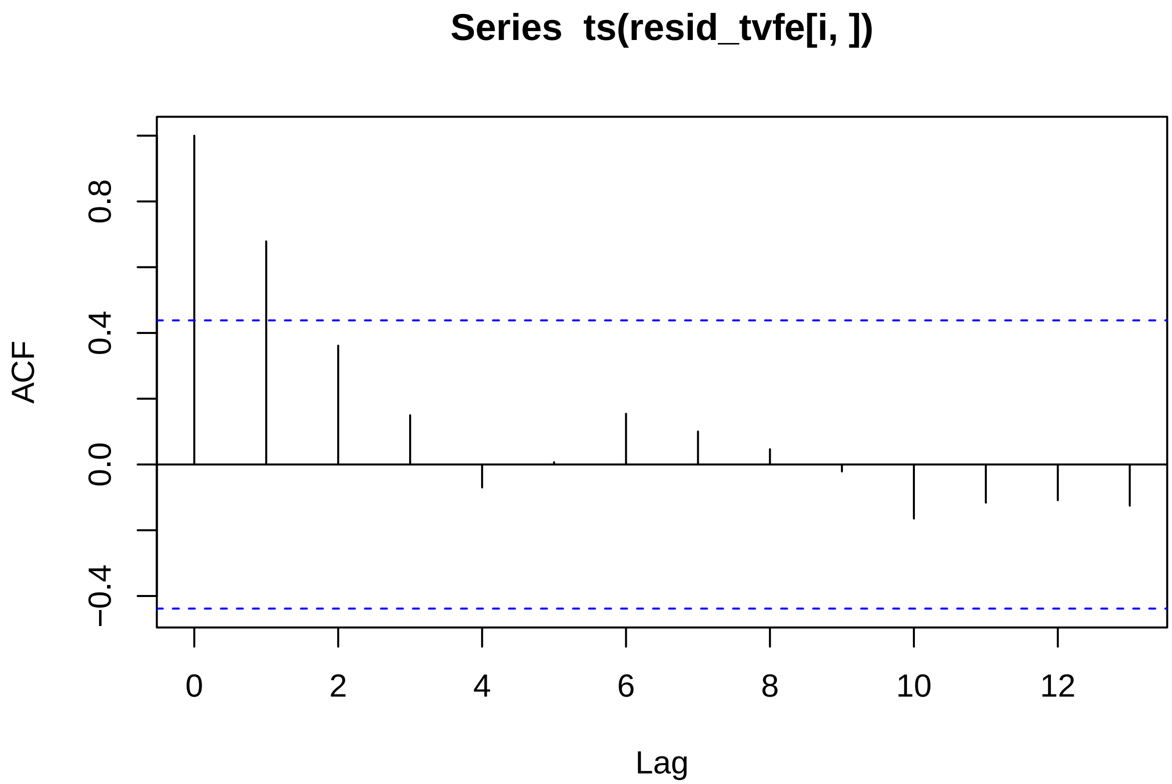
3) Residual's ACF

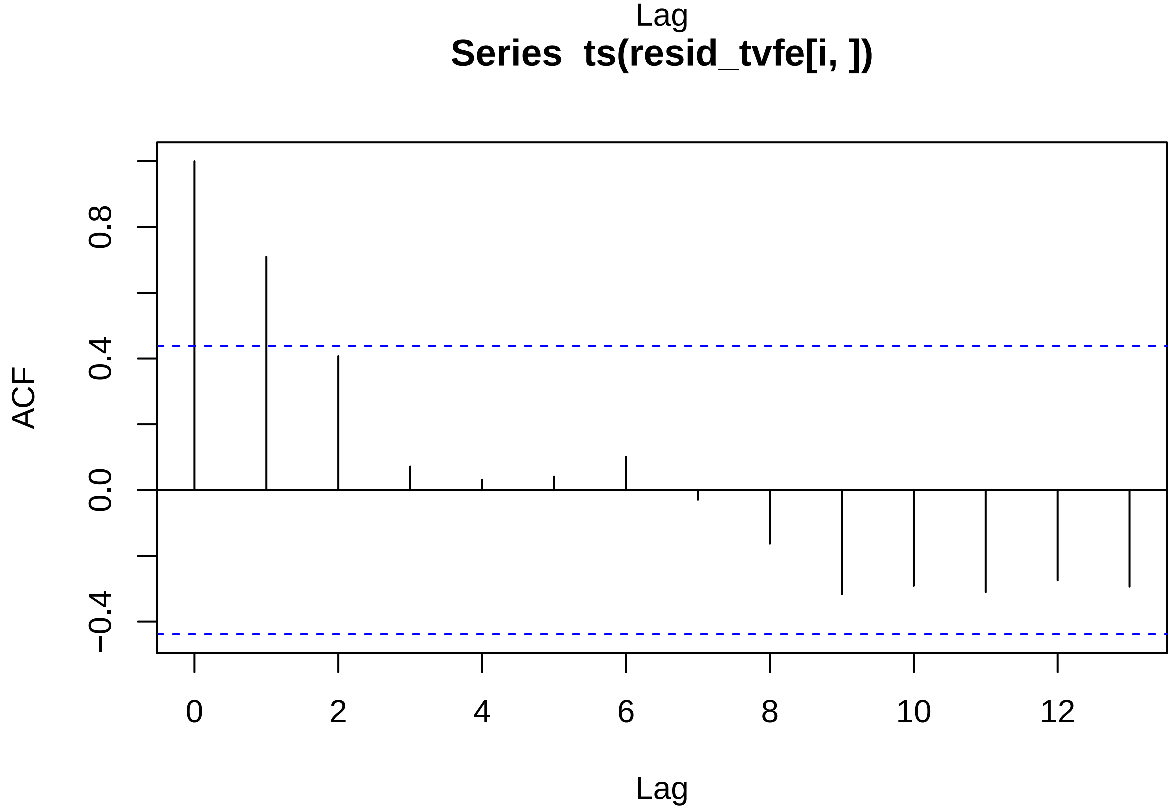
```
residtvfe <- mod.tvfe$residuals
residtvfe
```

```
##  [1] -0.0299086554 -0.0203957179 -0.0170282207 -0.0112842136 -0.0133602772
##  [6] -0.0132586930 -0.0131839386 -0.0086463223  0.0007326415 -0.0058545071
## [11] -0.0248844612 -0.0251151748 -0.0053563374  0.0172846576  0.0215196227
## [16]  0.0228276008  0.0069545267  0.0011758841  0.0061361314  0.0151714125
## [21]  0.0045219100  0.0080891285  0.0041243543  0.0009392622 -0.0010499586
## [26] -0.0054198398 -0.0025705788 -0.0077309952 -0.0033600487 -0.0038856282
## [31] -0.0228624526 -0.0122889704 -0.0160992703 -0.0132198268  0.0023445916
## [36] -0.0029286728 -0.0326472306 -0.0373523738 -0.0499753197 -0.0542166605
## [41]  0.0110555089  0.0142402626  0.0125747549  0.0178562644  0.0138250115
## [46]  0.0164165360  0.0192434470  0.0233899572  0.0284138120  0.0305436306
## [51]  0.0211416829  0.0220679435  0.0228928581  0.0292813989  0.0323452699
## [56]  0.0350254376  0.0268632764  0.0219465352  0.0158088280  0.0193215284
```

```
N=3
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```








```
pro_reject
```

```
##  [1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

4) 3-steps out-of-sample predition MSE (to forecast the log(CO2) value in year 2017,2018 and 2019)

TVFE:

```
mod.tvfe <- tvReg::tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                 data = e, method ="within", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.25
```

```
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
  forca <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)

MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}

MSE
```

```
## [1] 0.001379375 0.002512239 0.002772289
```

FE:

```
library(plm)
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech")
```

```
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech")
  forca =c(0,0,0)
  forca <- predict(mod.fe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.
```

```
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)

MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}

MSE
```

```
## [1] 0.002307142 0.004900446 0.005569003
```

#NW(5)

```
e_0915 <- read_excel("~/Desktop/RP/all_0915.xlsx", sheet = "NW", range = "C1:K116")
# name the vectors:
colnames(e_0915) <- c("province","year","lco2","lgdp","lgdp2","ff","tech","region","yearstd")
e <- filter(e_0915, year<=2016)
ff<-e$ff
tech<-e$tech
```

FE

```
mod.fe <- plm::plm(lco2~lgdp+lgdp2+ff+tech, index = c("province", "year"), model = "within", data=e)
mod.fe.CI <- confint(mod.fe, level = 0.68)
mod.fe.CI
```

```
##                 16 %         84 %
## lgdp   1.4526084380  1.867042407
```

```
## lgdp2 -0.2371570808 -0.090356806
## ff    -0.0004145384  0.000556731
## tech   0.0217188700  0.023427792
```

TVFE

Kernel: Gaussian Bandwidth: 0.6 Method: within

```
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                 data = e, method ="within", bw =NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.6747776
```

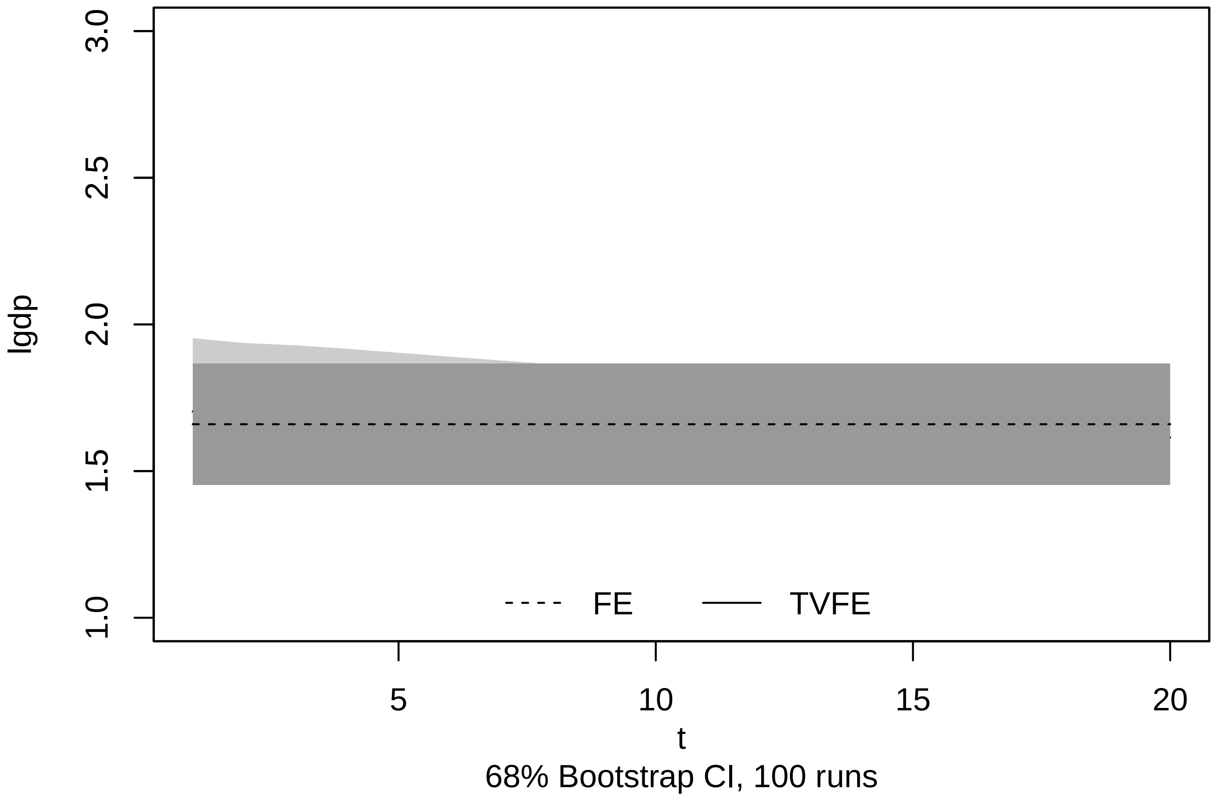
```
# Bootstrapping to get 95% confidence intervals
mod.tvfe.CI <- confint(mod.tvfe, level = 0.68)
mod.tvfe.CI
```

```
##
## Class:  tvplm
##
## Mean of coefficient estimates:
## ================================
##       lgdp      lgdp2         ff       tech    yearstd
##  1.6550529 -0.1615911  0.0001437  0.0225751 -1.8528670
##
## LOWER (68%):
##       lgdp      lgdp2         ff       tech    yearstd
##  1.4655445 -0.2226507 -0.0008442  0.0205062 -2.0479151
##
## UPPER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  1.844561 -0.100531  0.001132  0.024644 -1.657819
##
## Bandwidth:  0.6748
```

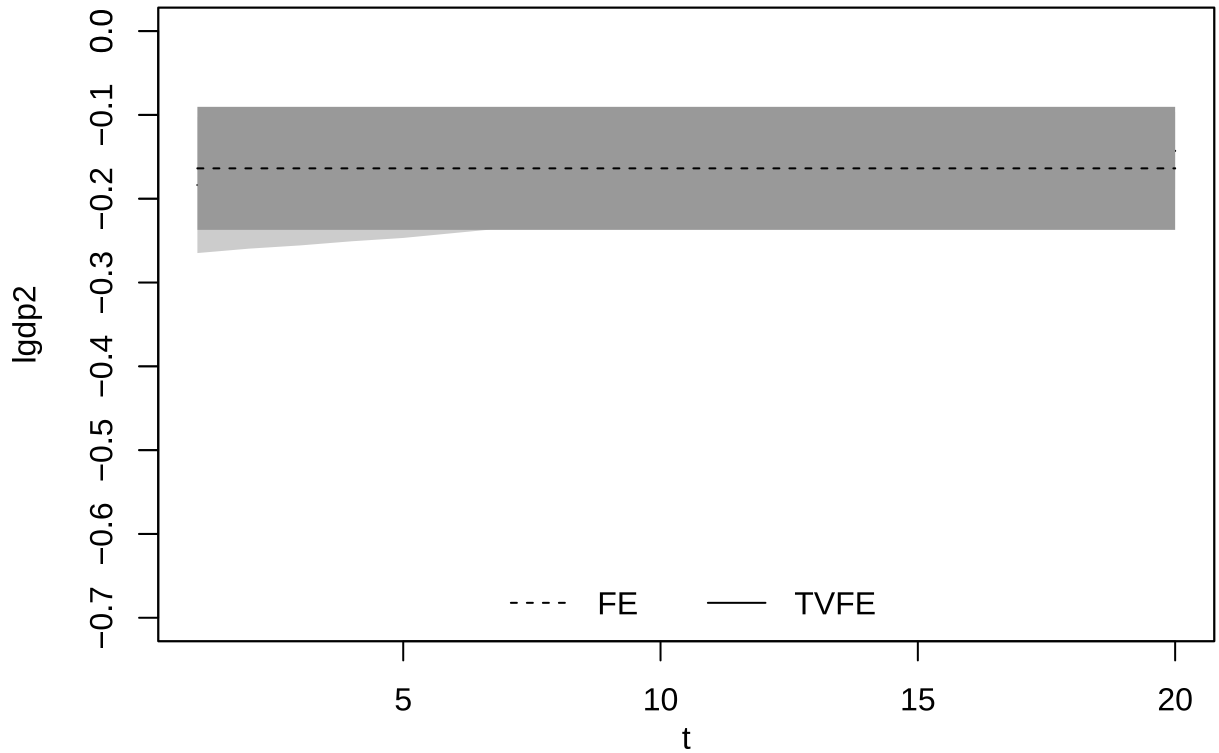
Post-estimation

1) Comparison plot of the within estimators

```
# see the time-varying pattern of the linear term's parameter:
plot(mod.tvfe.CI, vars = 1, ylim = c(1, 3))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), ylim = c(1, 3), ylab ="", xlab ="", type = "l",
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[1, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[1
lines(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```

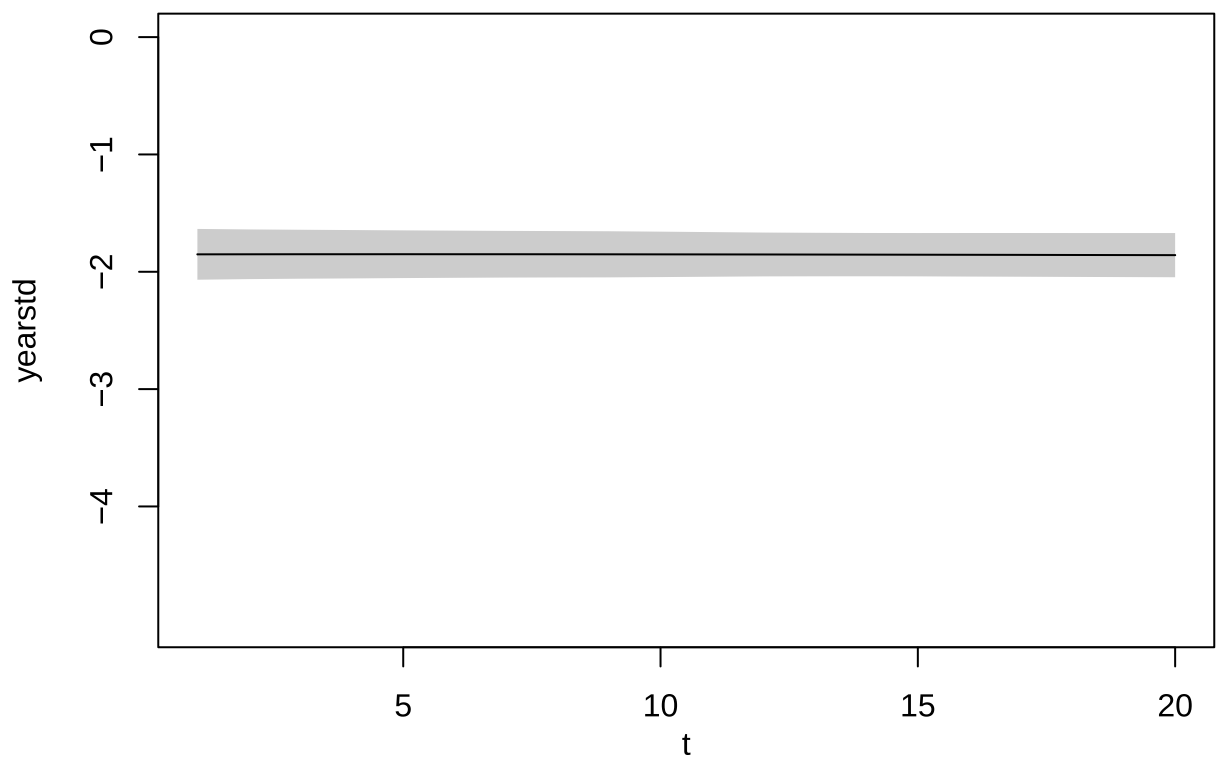



```r
# see the time-varying pattern of the quadratic term's parameter:
plot(mod.tvfe.CI, vars = 2, ylim = c(-0.7, 0))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), ylim = c(-0.7, 0), ylab ="", xlab ="", type = "
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[2, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[2
lines(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```



68% Bootstrap CI, 100 runs

```r
# see the estimated trend function:
plot(mod.tvfe.CI, vars = 5, ylim = c(-5, 0))
```



68% Bootstrap CI, 100 runs

```r
graphics::par(mfrow = c(1, 5),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
```

2) in-sample forecasting MSE

```r
fittedtvfe <- mod.tvfe$fitted
MSE_fe <- mean((mod.fe$residuals)^2)
MSE_fe
```

```
## [1] 0.008026555
```

```r
MSE_tvfe <- mean((mod.tvfe$residuals)^2)
MSE_tvfe
```

```
## [1] 0.02665344
```

3) Residual's ACF

```r
residtvfe <- mod.tvfe$residuals
residtvfe
```

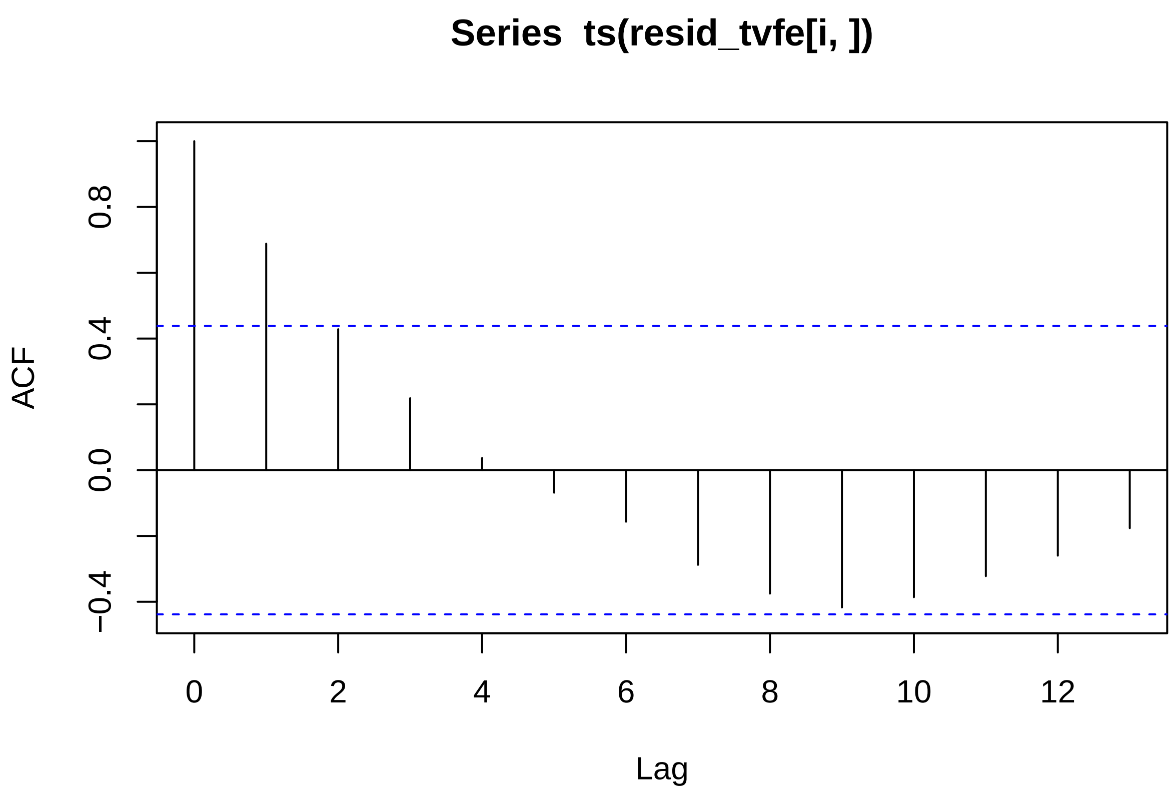
```
##   [1] -0.025649551  0.026621739  0.067852967  0.079334443  0.092298947
##   [6]  0.089254843  0.075306804  0.087527956  0.101617665  0.114558112
##  [11]  0.113176438  0.107453987  0.101939106  0.084176781  0.068024139
##  [16]  0.052897688  0.033108679  0.014914963  0.013390377 -0.005148524
##  [21] -0.388410179 -0.472478602 -0.327671327 -0.418992616 -0.459854780
##  [26] -0.460466258 -0.136803974 -0.419090157 -0.491480885 -0.386893187
##  [31] -0.306012547 -0.205453673 -0.277950374 -0.163227315 -0.172785322
##  [36] -0.109110671 -0.061839336 -0.051428767 -0.037805933 -0.011906394
##  [41]  0.135351205  0.163037985  0.119899964  0.187370039  0.148878245
##  [46]  0.148938010  0.137524783  0.133924888  0.127086090  0.111979918
##  [51]  0.094540532  0.078834959  0.073642351  0.050186066  0.048785116
##  [56]  0.044395163  0.041209490  0.029004350 -0.038959519 -0.030196821
##  [61]  0.143781536  0.171392756  0.211896807  0.226100841  0.216286762
##  [66]  0.193236909  0.177191691  0.147430365  0.054991821  0.095968267
##  [71]  0.075419595  0.065937658  0.047116684  0.026633662  0.004548666
##  [76] -0.002815270 -0.004004593 -0.015294827 -0.010160208 -0.004348759
##  [81] -0.088947457 -0.075981534 -0.016605365  0.035015423  0.045293117
##  [86]  0.073683758  0.070273227  0.050937296  0.055270087  0.055871879
##  [91]  0.061425193  0.050216777  0.033266064  0.034065723  0.029354838
##  [96]  0.026795698  0.016401072  0.013174024  0.018482824  0.021296427
```

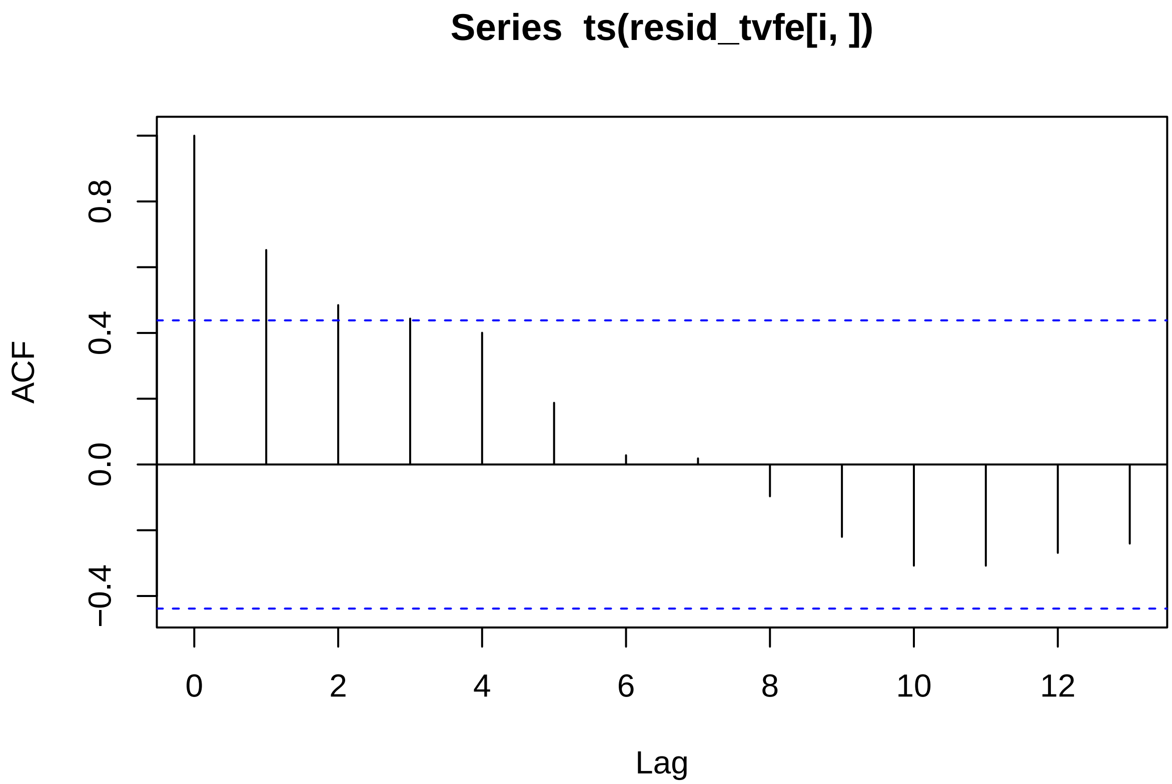
```r
N=5
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
```

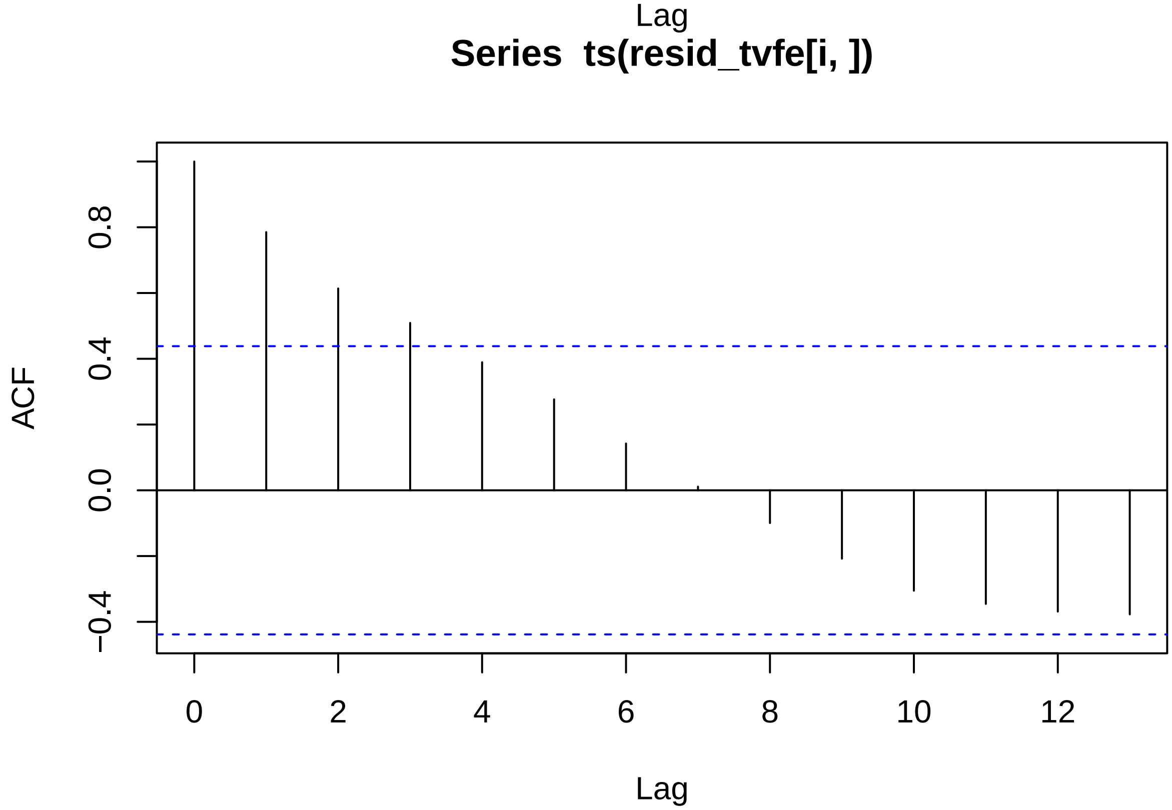
```
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```

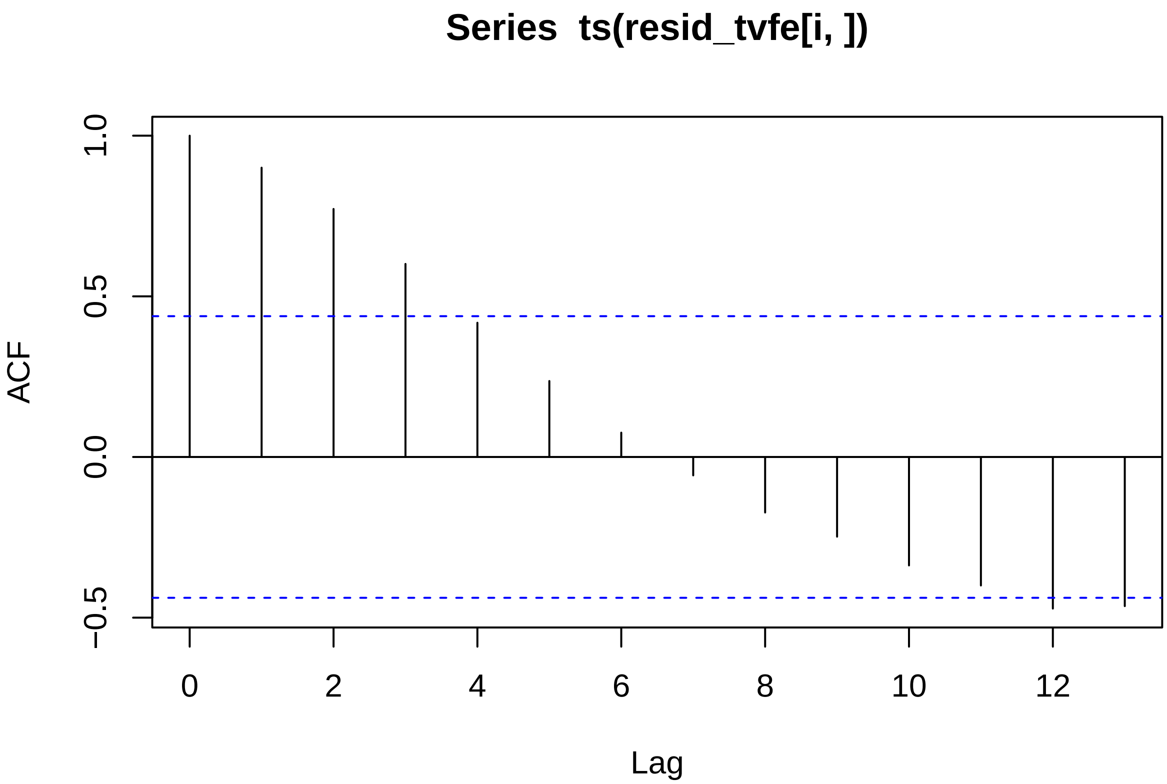
```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value
```

```
## Warning in kpss.test(ts(data_i)): p-value greater than printed p-value
```

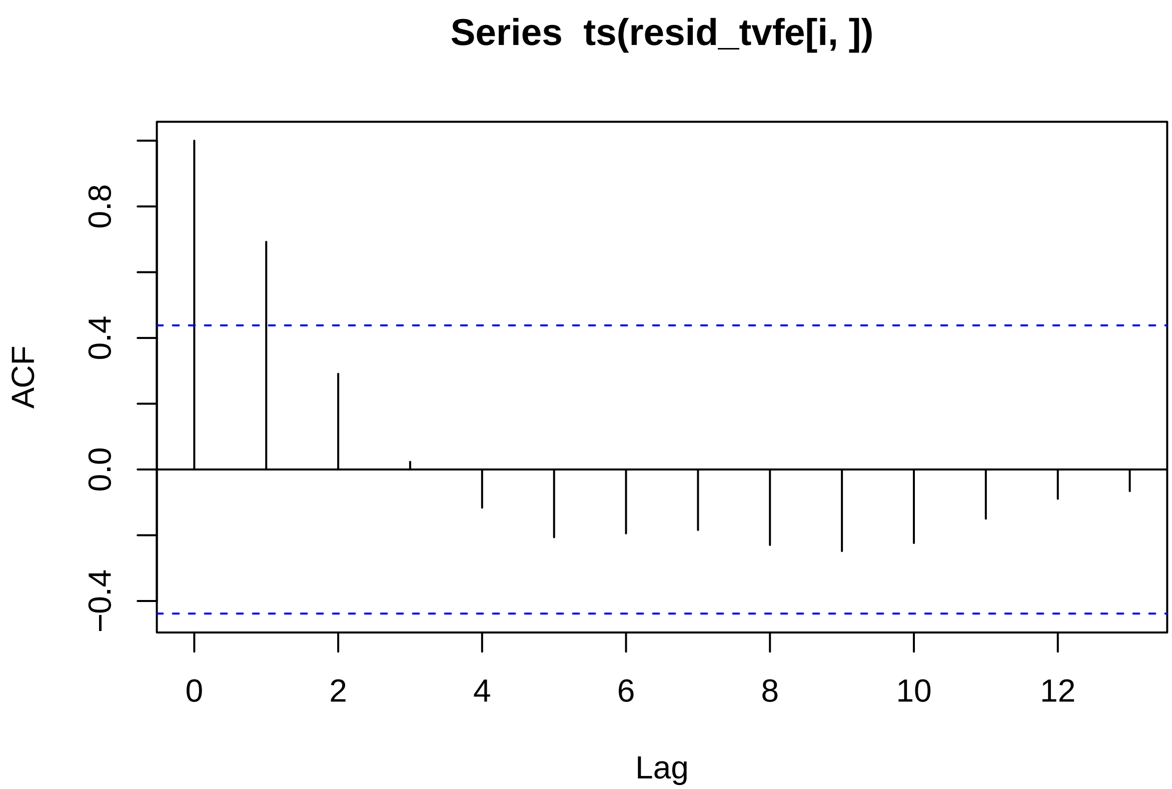



Series ts(resid_tvfe[i,])
ACF
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12
Lag


Series ts(resid_tvfe[i,])
ACF
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12
Lag




```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value

## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value
```




```
pro_reject
```

```
##  [1] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

4) 3-steps out-of-sample predition MSE (to forecast the log(CO2) value in year 2017,2018 and 2019)

TVFE:

```
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                 data = e, method ="pooling", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  20
```

```
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
```

```r
  force <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 5, ncol = 3, byrow = TRUE)

MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}

MSE
```

```
## [1] 0.004524210 0.007135934 0.010536268
```

FE:

```r
library(plm)
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech")
  forca =c(0,0,0)
  forca <- predict(mod.fe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
```

```
## original model used for prediction, see ?predict.plm.
```

```
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)
```

```
## Warning in matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE): data length
## differs from size of matrix: [15 != 3 x 3]
```

```
MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}
```

```
## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length
```

```
MSE
```

```
## [1] 0.05143450 0.05968609 0.06455821
```

#S (3)

```
e_0915 <- read_excel("~/Desktop/RP/all_0915.xlsx", sheet = "S", range = "C1:K70")
# name the vectors:
colnames(e_0915) <- c("province","year","lco2","lgdp","lgdp2","ff","tech","region","yearstd")
e <- filter(e_0915, year<=2016)
ff<-e$ff
tech<-e$tech
```

FE

```
mod.fe <- plm::plm(lco2~lgdp+lgdp2+ff+tech, index = c("province", "year"), model = "within", data=e)
mod.fe.CI <- confint(mod.fe, level = 0.68)
mod.fe.CI
```

```
##                16 %         84 %
## lgdp   1.665050754  2.043601479
## lgdp2 -0.318841984 -0.195297550
## ff    -0.008225814 -0.003071381
## tech   0.066397277  0.077744477
```

TVFE

Kernel: Gaussian Bandwidth: 0.6 Method: within

```
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                  data = e, method ="within", bw =NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  1
```

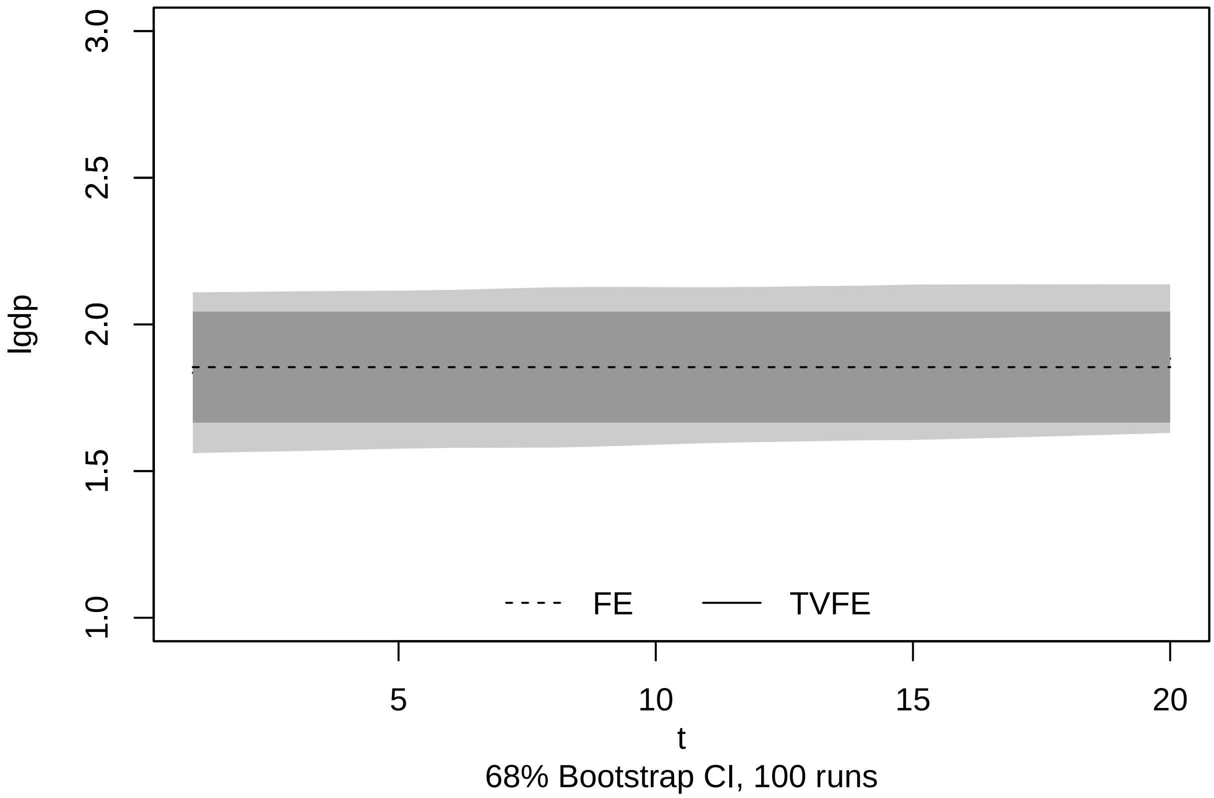
```
# Bootstrapping to get 95% confidence intervals
mod.tvfe.CI <- confint(mod.tvfe, level = 0.68)
mod.tvfe.CI
```

```
##
## Class:  tvplm
##
## Mean of coefficient estimates:
## ================================
##      lgdp     lgdp2        ff      tech   yearstd
##  1.859568 -0.257853 -0.005931  0.072750 -2.337054
##
## LOWER (68%):
##     lgdp    lgdp2       ff     tech  yearstd
##  1.59311 -0.30882 -0.01318  0.04743 -2.77368
##
## UPPER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  2.126026 -0.206881  0.001322  0.098067 -1.900426
##
## Bandwidth:  1
```

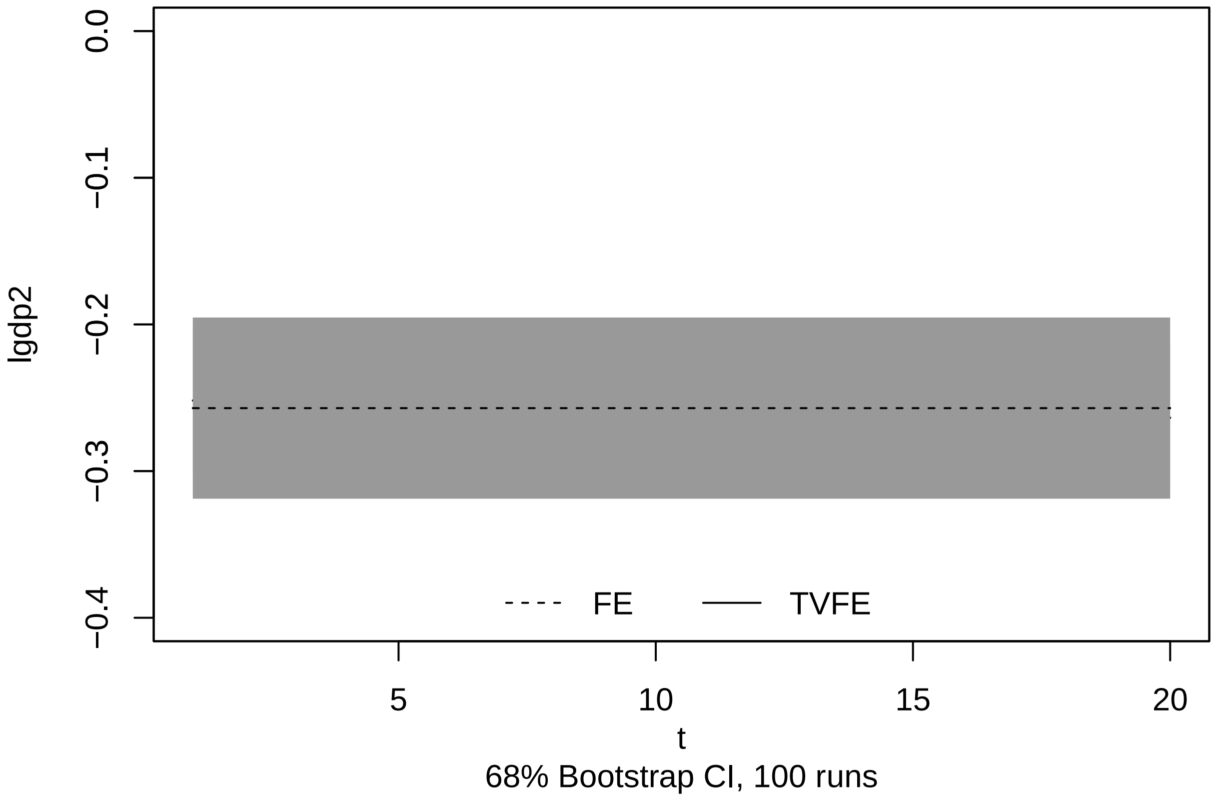
Post-estimation

1) Comparison plot of the within estimators

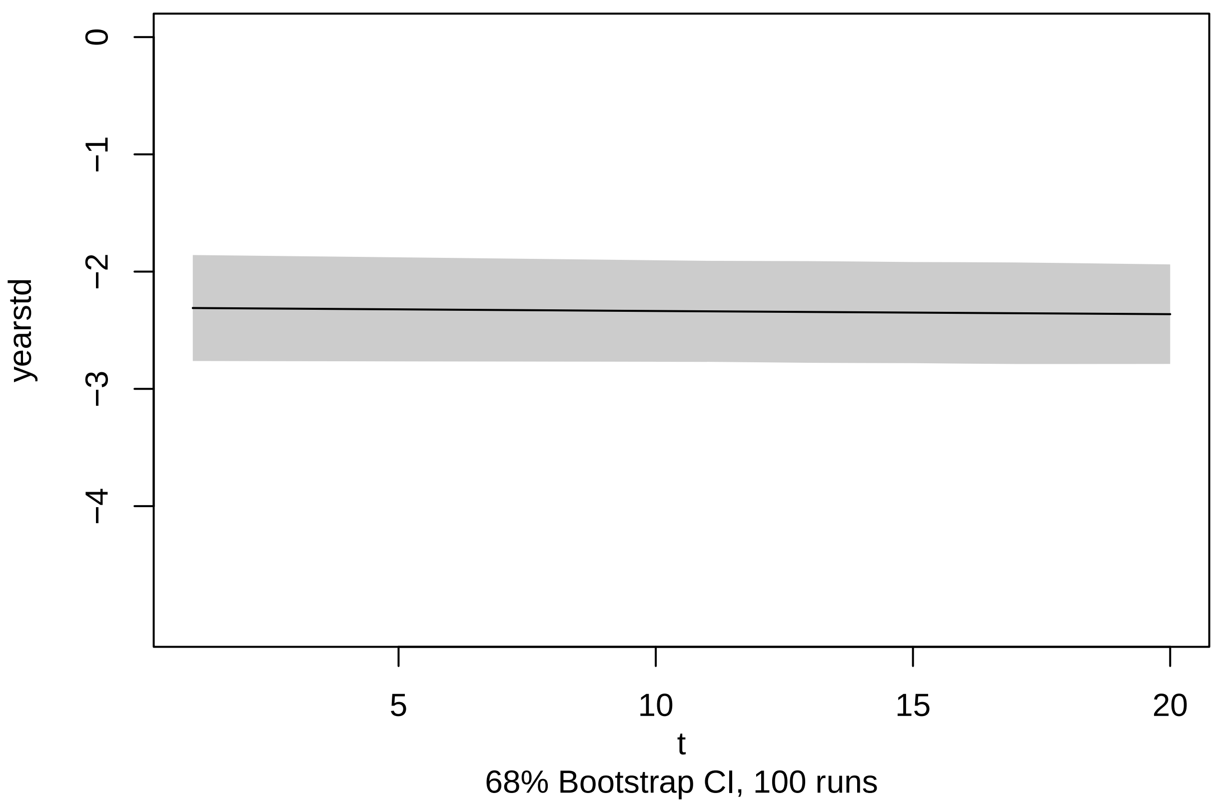
```
# see the time-varying pattern of the linear term's parameter:
plot(mod.tvfe.CI, vars = 1, ylim = c(1, 3))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), ylim = c(1, 3), ylab ="", xlab ="", type = "l",
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[1, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[1
lines(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```
# see the time-varying pattern of the quadratic term's parameter:
plot(mod.tvfe.CI, vars = 2, ylim = c(-0.4, 0))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), ylim = c(-0.4, 0), ylab ="", xlab ="", type = "
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[2, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[2
lines(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```
# see the estimated trend function:
plot(mod.tvfe.CI, vars = 5, ylim = c(-5, 0))
```



```
graphics::par(mfrow = c(1, 5),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
```

2) in-sample forecasting MSE

```
fittedtvfe <- mod.tvfe$fitted
MSE_fe <- mean((mod.fe$residuals)^2)
MSE_fe
```

```
## [1] 0.004076715
```

```
MSE_tvfe <- mean((mod.tvfe$residuals)^2)
MSE_tvfe
```

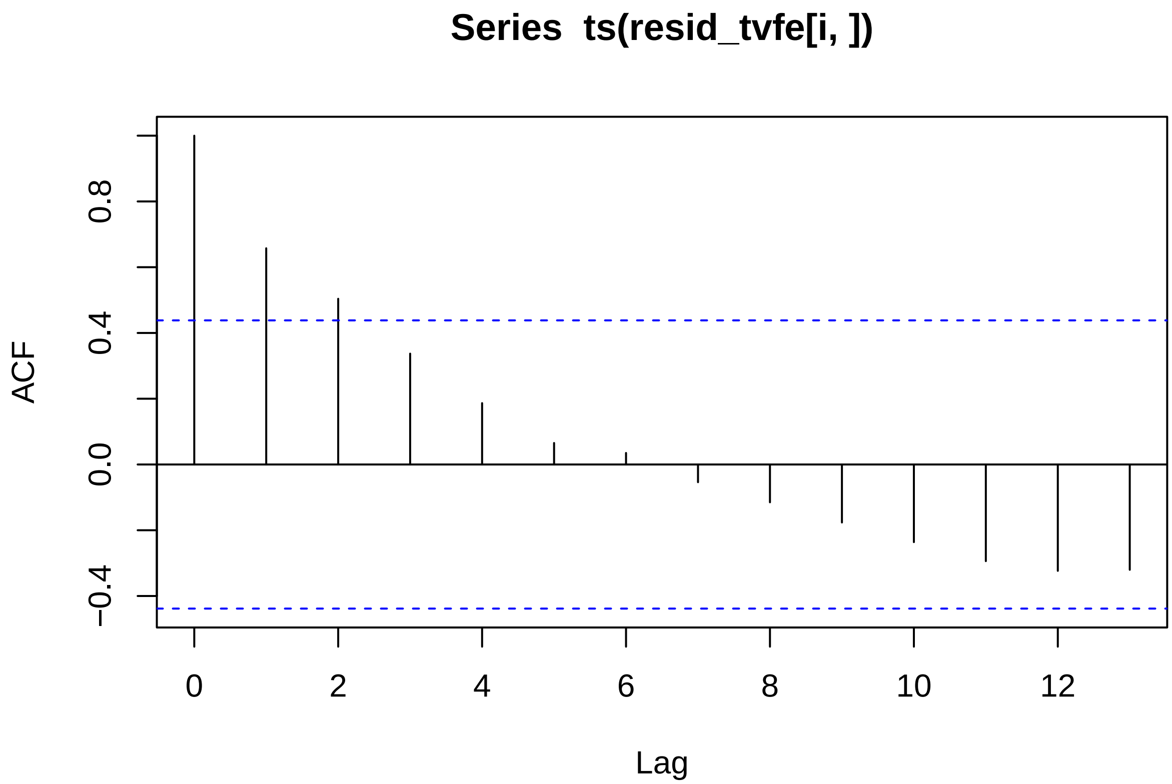
```
## [1] 0.004066003
```

3) Residual's ACF

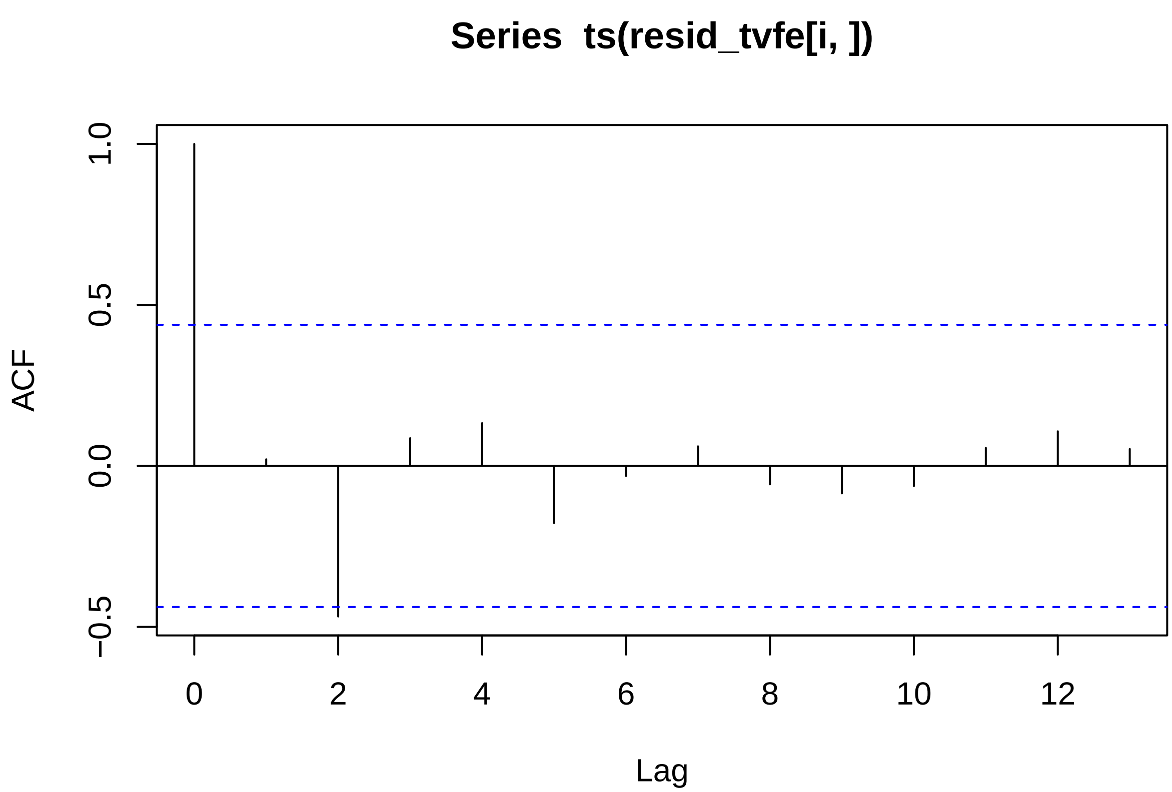
```
residtvfe <- mod.tvfe$residuals
residtvfe
```

```
##  [1] -0.022739860  0.028829410  0.023887472  0.027637618  0.031244478
##  [6]  0.030634113  0.021998394  0.027045102  0.022100889  0.016187641
## [11]  0.007457731  0.007975626  0.007633031  0.006402867  0.005348936
## [16] -0.005281244 -0.013816642 -0.023199383 -0.033476085 -0.043695156
## [21] -0.041845456  0.001543504  0.029208482 -0.066055494  0.001265957
## [26]  0.053176335  0.022453034 -0.038556205 -0.018097463 -0.016679480
## [31] -0.006180287 -0.001413637 -0.006810277 -0.012599477 -0.014229875
## [36] -0.011170106  0.009763683  0.019831480  0.013461623  0.012997750
## [41]  0.119531618 -0.099187172  0.108659163  0.075700206  0.055079599
## [46] -0.384200628  0.032861712  0.017806515  0.002907347  0.025142429
## [51] -0.108361379 -0.063831101 -0.028338296 -0.016915372 -0.002448776
## [56]  0.021844369  0.034896186  0.039133745  0.040290331  0.058735015
```

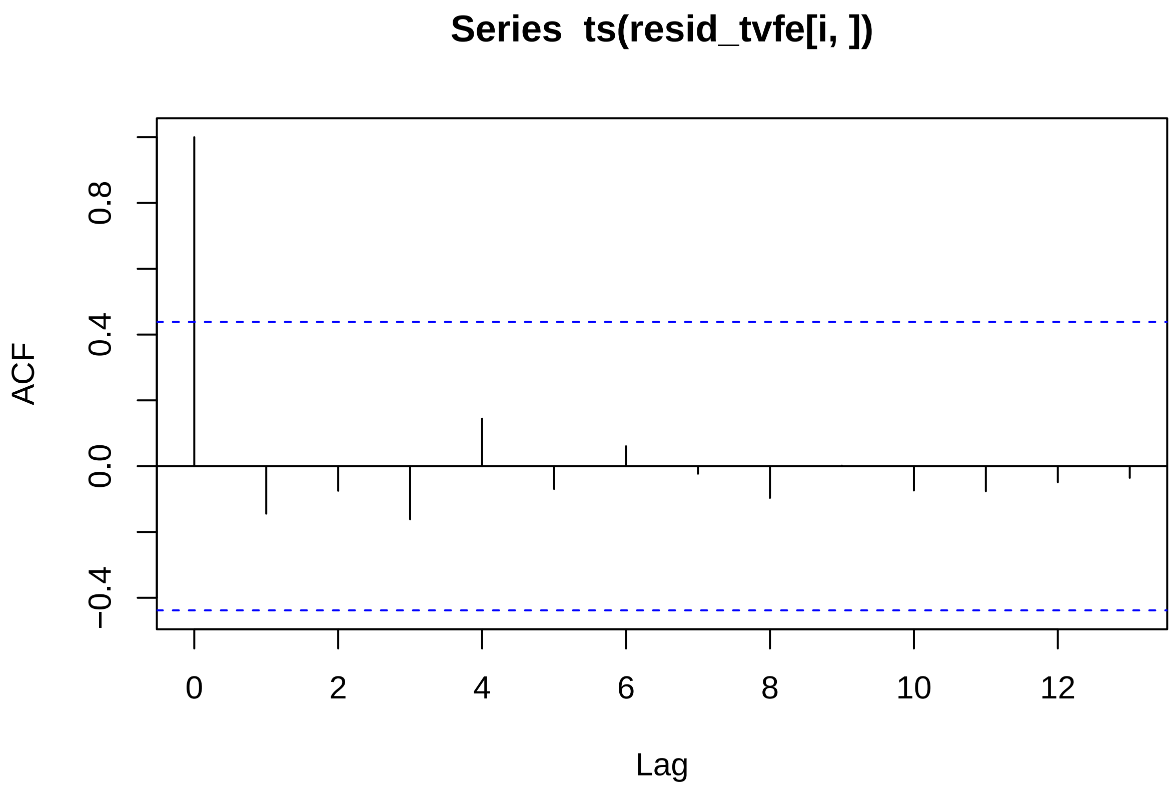
```
N=3
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```




```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value

## Warning in kpss.test(ts(data_i)): p-value greater than printed p-value
```



```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value

## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value
```




```
pro_reject
```

```
##  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

4) 3-steps out-of-sample predition MSE (to forecast the log(CO2) value in year 2017,2018 and 2019)

TVFE:

```
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                  data = e, method ="pooling", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  20
```

```
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
```

```
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
  forca <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)

MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}

MSE
```

```
## [1] 0.001940786 0.002255282 0.002675460
```

FE:

```
library(plm)
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech")
  forca =c(0,0,0)
  forca <- predict(mod.fe, newdata = new_data, n.ahead = 3) #1x3
  forca_matrix <-rbind(forca_matrix,forca) # p x 3 matrix
}
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.
```

```
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)

MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}

MSE
```

```
## [1] 0.001898252 0.002233523 0.002652738
```

SW (4)

```
e_0915 <- read_excel("~/Desktop/RP/all_0915.xlsx", sheet = "Western", range = "C1:K93")
# name the vectors:
colnames(e_0915) <- c("province","year","lco2","lgdp","lgdp2","ff","tech","region","yearstd")
e <- filter(e_0915, year<=2016)
ff<-e$ff
tech<-e$tech
```

FE

```
mod.fe <- plm::plm(lco2~lgdp+lgdp2+ff+tech, index = c("province", "year"), model = "within", data=e)
mod.fe.CI <- confint(mod.fe, level = 0.68)
mod.fe.CI
```

```
##               16 %        84 %
## lgdp   0.599367074 1.025444954
## lgdp2 -0.108427481 0.034807273
## ff     0.002151271 0.006863931
## tech   0.006812250 0.013935096
```

TVFE

Kernel: Gaussian Bandwidth: 0.6 Method: within

```
mod.tvfe <- tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                  data = e, method ="within", bw =NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.2727968
```

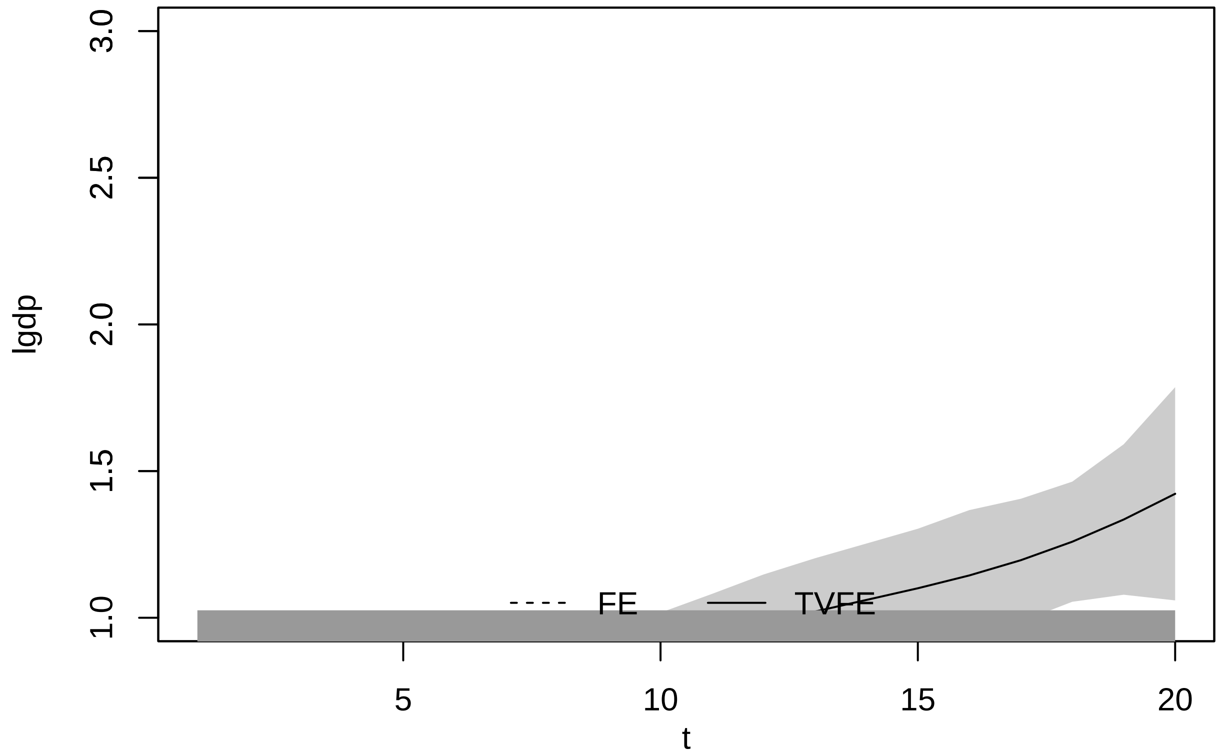
```r
# Bootstrapping to get 95% confidence intervals
mod.tvfe.CI <- confint(mod.tvfe, level = 0.68)
mod.tvfe.CI
```

```
##
## Class:  tvplm
##
## Mean of coefficient estimates:
## ===============================
##      lgdp     lgdp2        ff      tech   yearstd
##  0.879507 -0.037325  0.004679  0.012406 -0.881370
##
## LOWER (68%):
##       lgdp      lgdp2         ff       tech    yearstd
##  0.6720411 -0.1087699  0.0001528  0.0017976 -1.2606533
##
## UPPER (68%):
##      lgdp     lgdp2        ff      tech   yearstd
##  1.086973  0.034119  0.009205  0.023014 -0.502086
##
## Bandwidth:  0.2728
```

Post-estimation

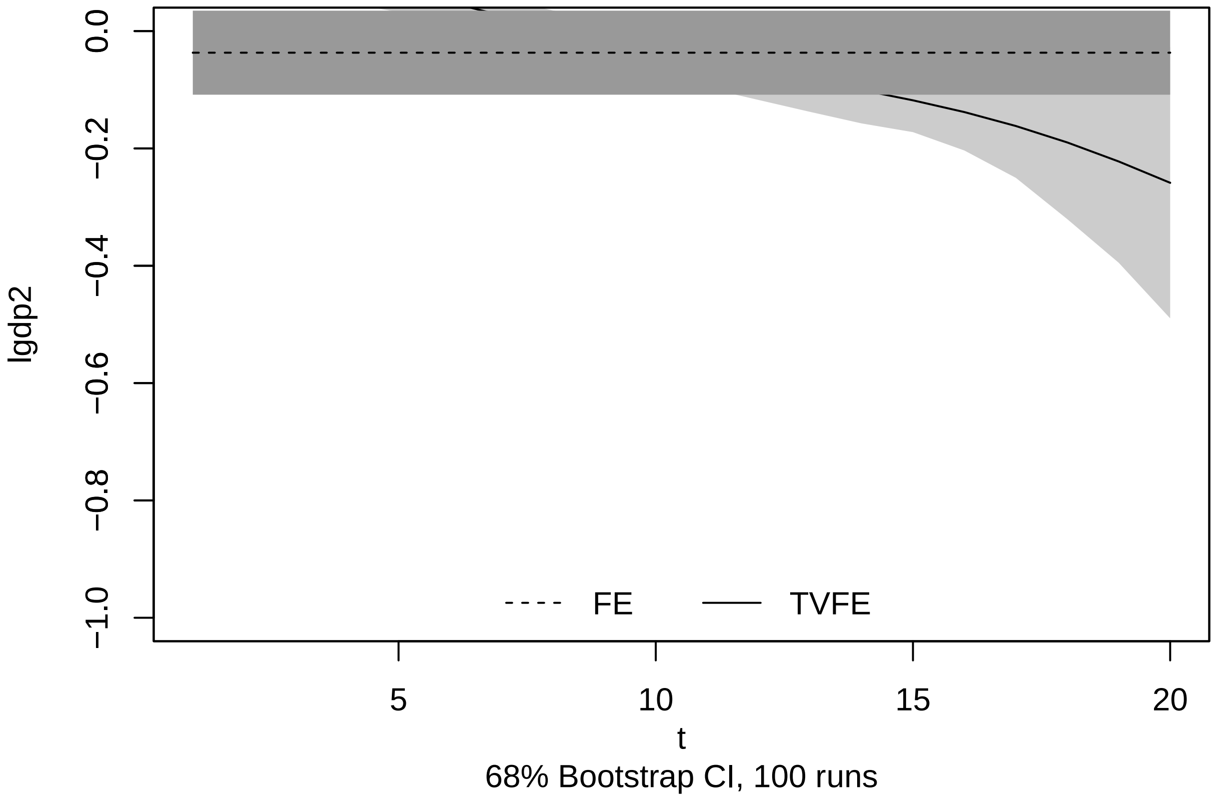
1) Comparison plot of the within estimators

```r
# see the time-varying pattern of the linear term's parameter:
plot(mod.tvfe.CI, vars = 1, ylim = c(1, 3))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), ylim = c(1, 3), ylab ="", xlab ="", type = "l",
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[1, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[1
lines(x.axis, rep(mean(mod.fe.CI[1,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```

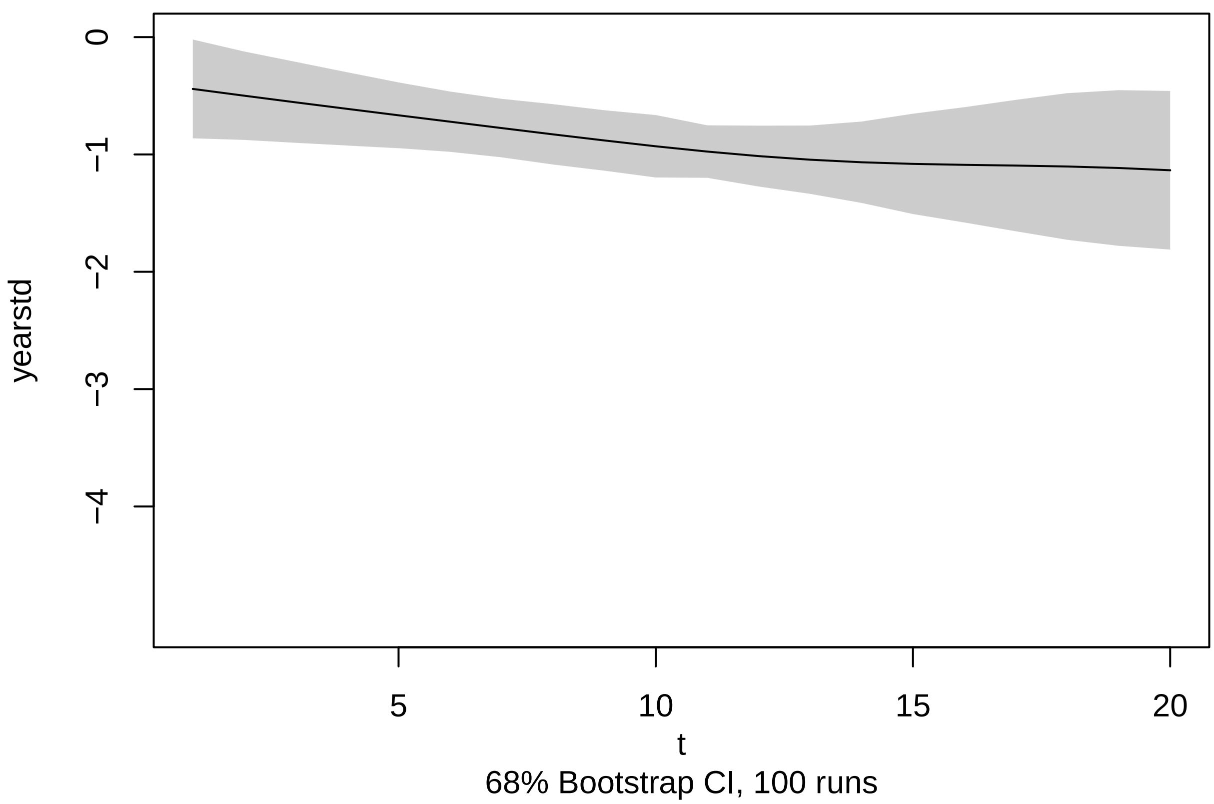


68% Bootstrap CI, 100 runs

```
# see the time-varying pattern of the quadratic term's parameter:
plot(mod.tvfe.CI, vars = 2, ylim = c(-1, 0))
graphics::par(mfrow = c(1, 1),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
par(new = TRUE)
plot(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), ylim = c(-1, 0), ylab ="", xlab ="", type = "l"
graphics::polygon(c(rev(x.axis), x.axis), c(rep(rev(mod.fe.CI[2, 2]), mod.tvfe.CI$obs), rep(mod.fe.CI[2
lines(x.axis, rep(mean(mod.fe.CI[2,]), mod.tvfe.CI$obs), lty=2)
legend("bottom", c("FE", "TVFE"), lty = 2:1, col = 1, ncol = 2, bty = "n")
```




```
# see the estimated trend function:
plot(mod.tvfe.CI, vars = 5, ylim = c(-5, 0))
```



```r
graphics::par(mfrow = c(1, 5),
              mar = c(4, 4, 2, 1), oma = c(0, 0, 0, 0))
x.axis <- 1:mod.tvfe.CI$obs
```

2) in-sample forecasting MSE

```r
fittedtvfe <- mod.tvfe$fitted
MSE_fe <- mean((mod.fe$residuals)^2)
MSE_fe
```

```
## [1] 0.007042975
```

```r
MSE_tvfe <- mean((mod.tvfe$residuals)^2)
MSE_tvfe
```

```
## [1] 0.01378975
```

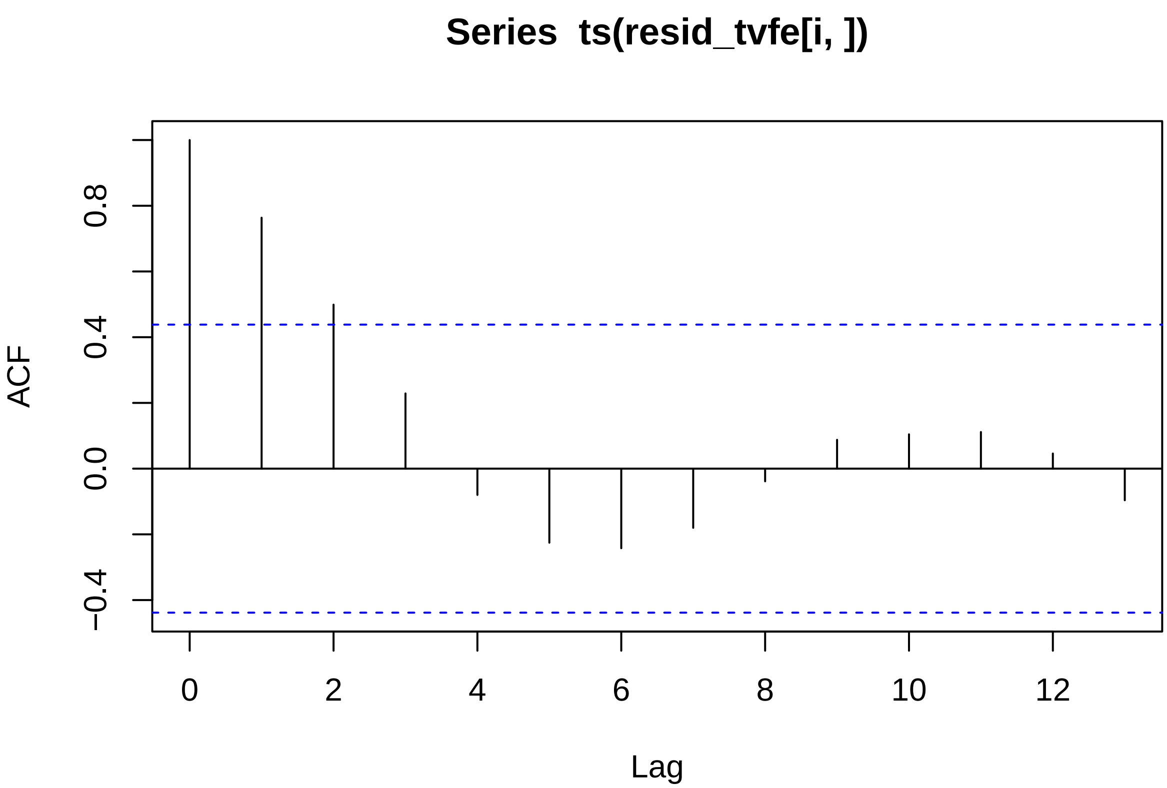
3) Residual's ACF

```r
residtvfe <- mod.tvfe$residuals
residtvfe
```

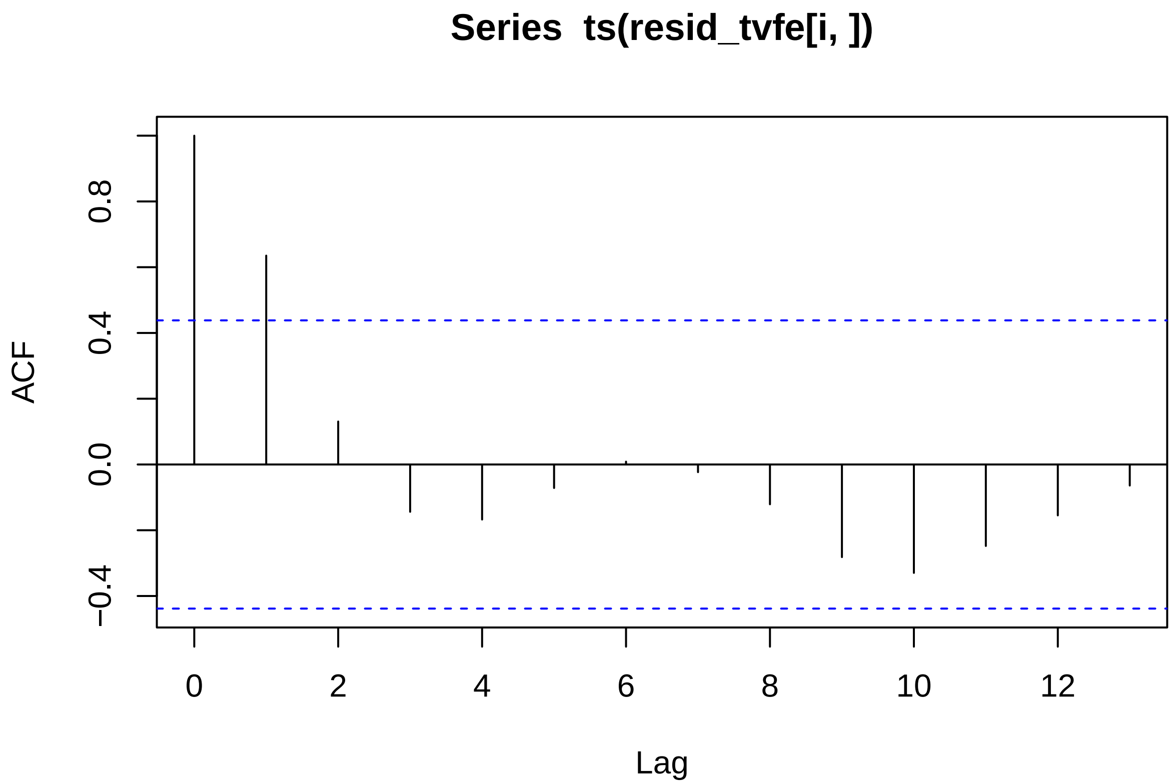
```
##  [1]  0.068173201  0.060703815  0.061076724  0.059428474  0.034853044
##  [6]  0.030344591 -0.006332164  0.008595625  0.020926004  0.026932092
## [11]  0.015097239  0.060673508  0.071088513  0.034611295  0.019182403
## [16] -0.005401573 -0.072819050 -0.077110449 -0.088761333 -0.093824091
## [21] -0.120258118 -0.122779702 -0.123714831 -0.070857152 -0.095328390
## [26] -0.130722282 -0.095646239 -0.042410527 -0.051454724 -0.050637729
## [31] -0.067143322 -0.079578438 -0.070420712 -0.050772948 -0.019381908
## [36] -0.013501185 -0.057369822 -0.104416240 -0.147880693 -0.155149476
## [41]  0.062553061  0.114696453  0.056265002  0.079046169  0.059137241
## [46]  0.072285492  0.061208831  0.054706000  0.076958422  0.104957339
## [51]  0.110235827  0.130345540  0.140797818  0.174049974  0.190362762
## [56]  0.245254533  0.259389419  0.293056168  0.354481934  0.414032101
## [61]  0.010021401  0.009118759 -0.050968398 -0.044581691 -0.052926272
## [66] -0.039468577  0.004274792  0.007307659 -0.027491072 -0.049685448
## [71] -0.030015858 -0.023048099 -0.010458234 -0.033835349 -0.080569283
## [76] -0.097841785 -0.147533366 -0.145157942 -0.225222153 -0.254766731
```

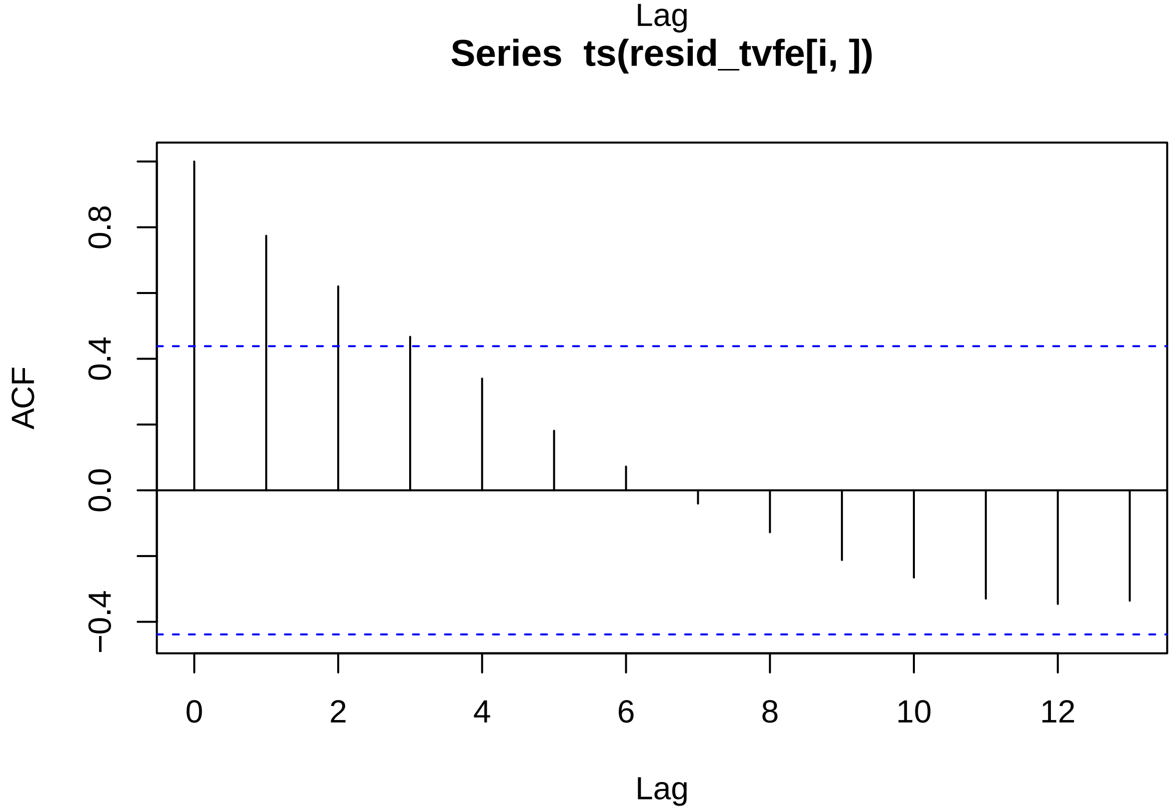
```r
N=4
# define a matrix to contain TVFE residules:
resid_tvfe= matrix(NA,nrow=N,ncol=20)
# define a vector to contain KPSS test results:
pro_reject <- logical(30)
for (i in (1:N)){
  resid_tvfe[i,] <- residtvfe[(1+20*(i-1)):(20*i)]
  acf(ts(resid_tvfe[i,]))
```

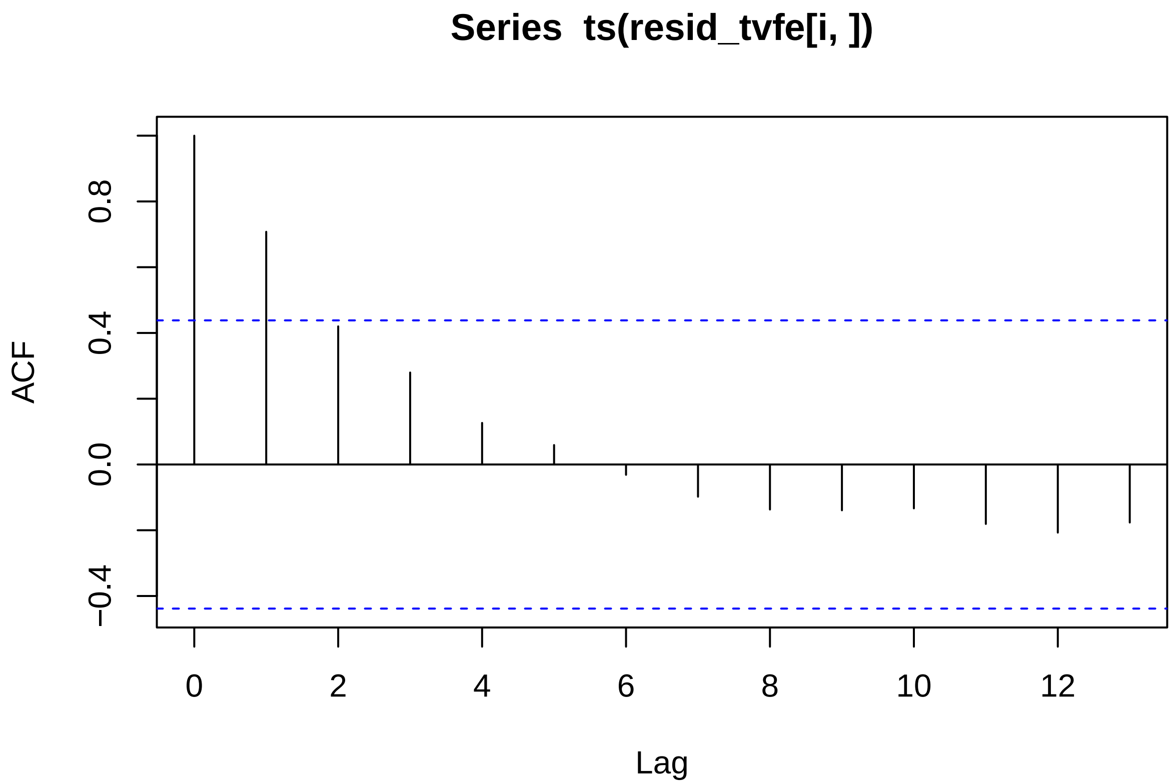
```
  # do a kpss test
  data_i = ts(resid_tvfe[i,])
  kpss.test(ts(data_i), null = c("Level"), lshort = TRUE)
  p_value <- kpss.test(ts(data_i))$p.value
  pro_reject[i] <- p_value < .05
}
```




```
## Warning in kpss.test(ts(data_i), null = c("Level"), lshort = TRUE): p-value
## greater than printed p-value

## Warning in kpss.test(ts(data_i)): p-value greater than printed p-value
```


Series ts(resid_tvfe[i,])
ACF
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12
Lag


Series ts(resid_tvfe[i,])
ACF
0.8
0.4
0.0
−0.4
0
2
4
6
8
10
12
Lag




```
pro_reject
```

```
##  [1]  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```

4) 3-steps out-of-sample predition MSE (to forecast the log(CO2) value in year 2017,2018 and 2019)

TVFE:

```
mod.tvfe <- tvReg::tvPLM(lco2~lgdp+lgdp2+ff+tech+yearstd, index = c("province", "year"),
                 data = e, method ="pooling", bw = NULL, tkernel="Gaussian")
```

```
## Calculating regression bandwidth... bw =  0.25
```

```
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
forca_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech","yearstd")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech","yearstd")
  forca =c(0,0,0)
```

```
  force <- forecast(mod.tvfe, newdata = new_data, n.ahead = 3) #1x3
  force_matrix <-rbind(force_matrix,force) # p x 3 matrix
}
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)
```

```
## Warning in matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE): data length
## differs from size of matrix: [12 != 3 x 3]
```

```
MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((force_matrix[,j]-observe_matrix[,j])^2)
}
```

```
## Warning in force_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in force_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in force_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length
```

```
MSE
```

```
## [1] 0.1944008 0.2061131 0.2217890
```

FE:

```
library(plm)
pro_list <- unique(e_0915$province)
year_list <- c(2017,2018,2019)
force_matrix <- c()
for(p in pro_list){ # loop p times
  select_data <- e_0915 %>%
    filter(province == p) %>%
    filter( year %in% year_list) %>%
    select( "lco2","lgdp","lgdp2","ff","tech")
  new_data <- select_data%>% select("lgdp","lgdp2","ff","tech")
  force =c(0,0,0)
  force <- predict(mod.fe, newdata = new_data, n.ahead = 3) #1x3
  force_matrix <-rbind(force_matrix,force) # p x 3 matrix
}
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.
```

```
## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.

## Warning in predict.plm(mod.fe, newdata = new_data, n.ahead = 3): Data supplied
## in argument 'newdata' is not a pdata.frame; weighted mean of fixed effects as in
## original model used for prediction, see ?predict.plm.
```

```
observe <- e_0915%>%
  filter(year %in% year_list) %>%
  select("lco2")
observe_matrix <- matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE)
```

```
## Warning in matrix(observe$lco2, nrow = 3, ncol = 3, byrow = TRUE): data length
## differs from size of matrix: [12 != 3 x 3]
```

```
MSE = c(0,0,0)

for (j in 1:3){
  MSE[j]= mean((forca_matrix[,j]-observe_matrix[,j])^2)
}
```

```
## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length

## Warning in forca_matrix[, j] - observe_matrix[, j]: longer object length is not
## a multiple of shorter object length
```

```
MSE
```

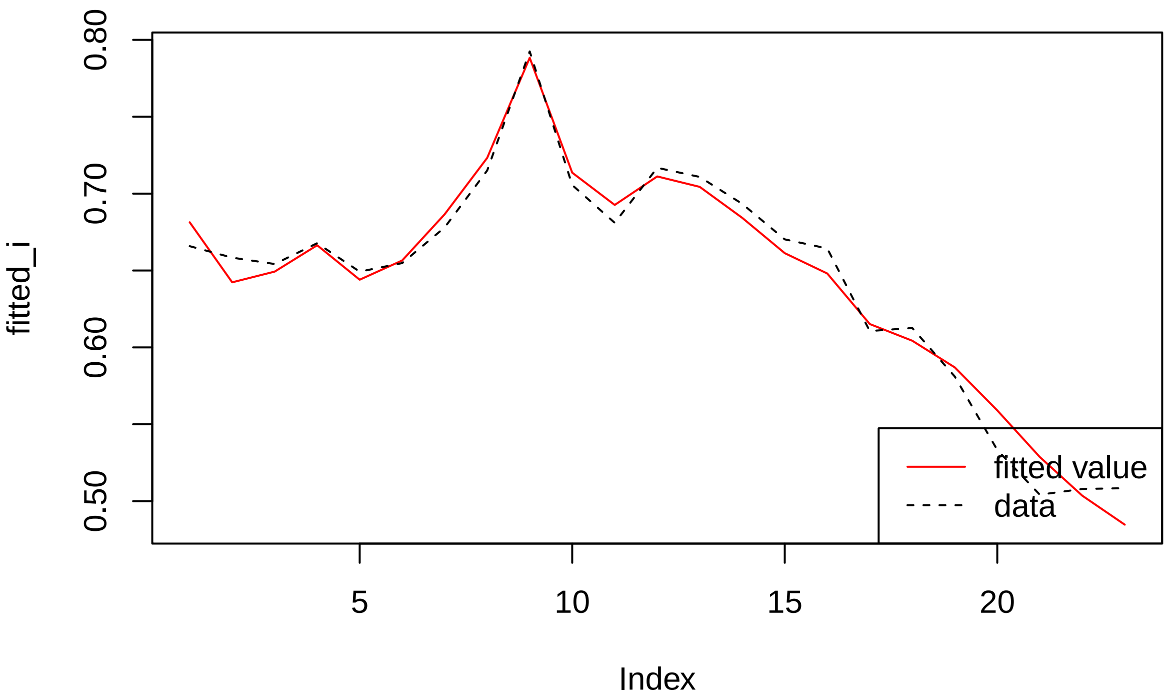
```
## [1] 0.1811883 0.1967218 0.2157243
```

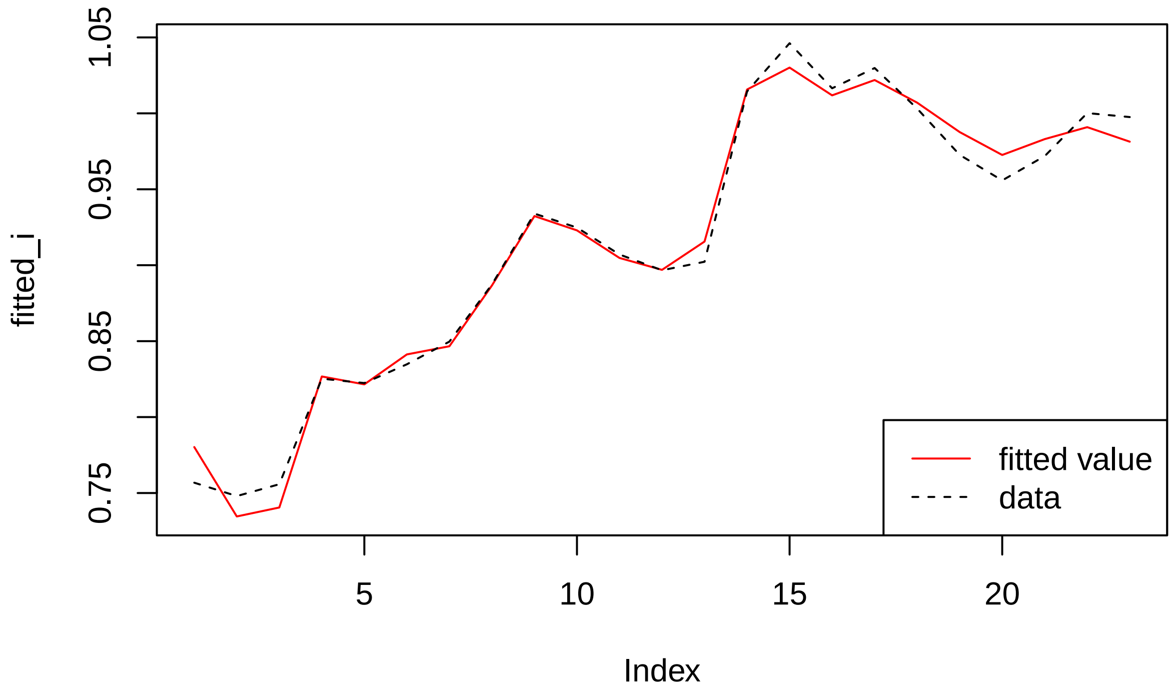
limitation and future direction

#Heterogeneity panel FE model with a second-order polynomial trend function

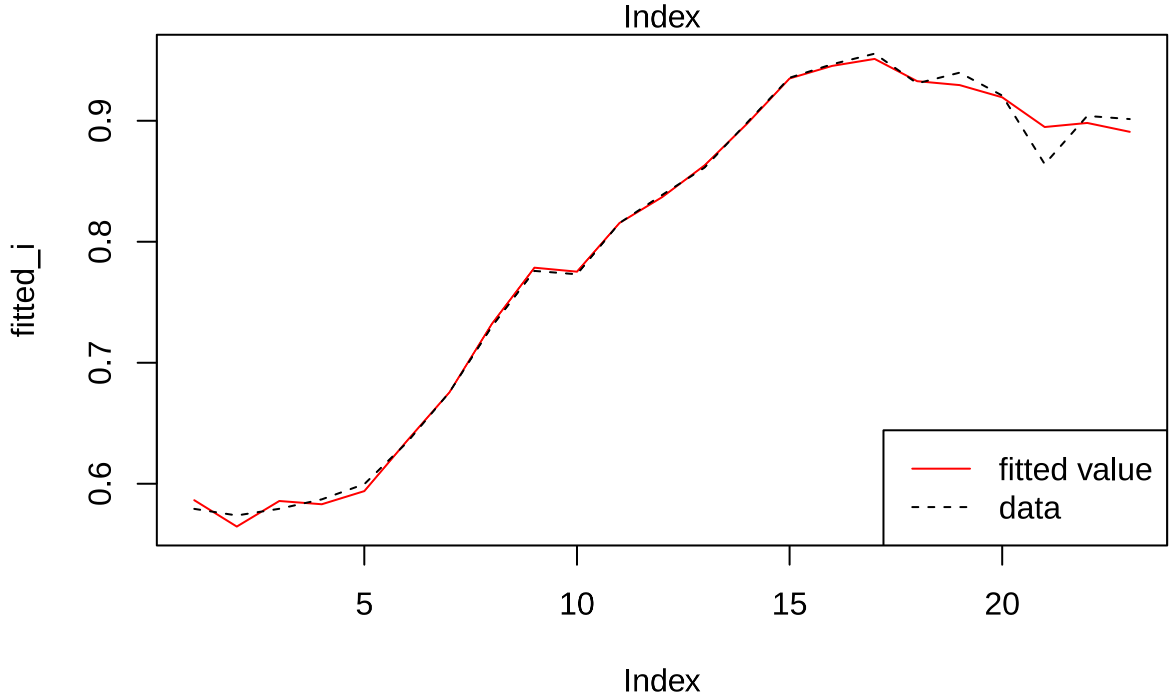
```
N=30
coefficient= matrix(NA,nrow=N,ncol=7)
#resid= matrix(NA,nrow=N,ncol=7)
for (i in (1:N)){
  data_i =  all_0915[all_0915["Index"]==i,c("lco2","lgdp","lgdp2","ff","tech","region")]
  year_std = (1:dim(data_i)[1])/dim(data_i)[1]
```

```
  est_i =  lm(lco2~lgdp+lgdp2+ff+tech+year_std + I(year_std^2),data=data_i)

  coefficient[i,] = est_i$coefficients
  residul_i=est_i$residuals
  fitted_i=est_i$fitted.values
  # residual_i
  #acf(residul_i)
  plot(fitted_i,ylim =c(min(data_i$lco2,fitted_i),max(data_i$lco2,fitted_i)),type='l',col='red')
  lines(data_i$lco2,lty=2)
  legend('bottomright',c('fitted value','data'),lty=c(1,2),col=c('red','black'))
}
```

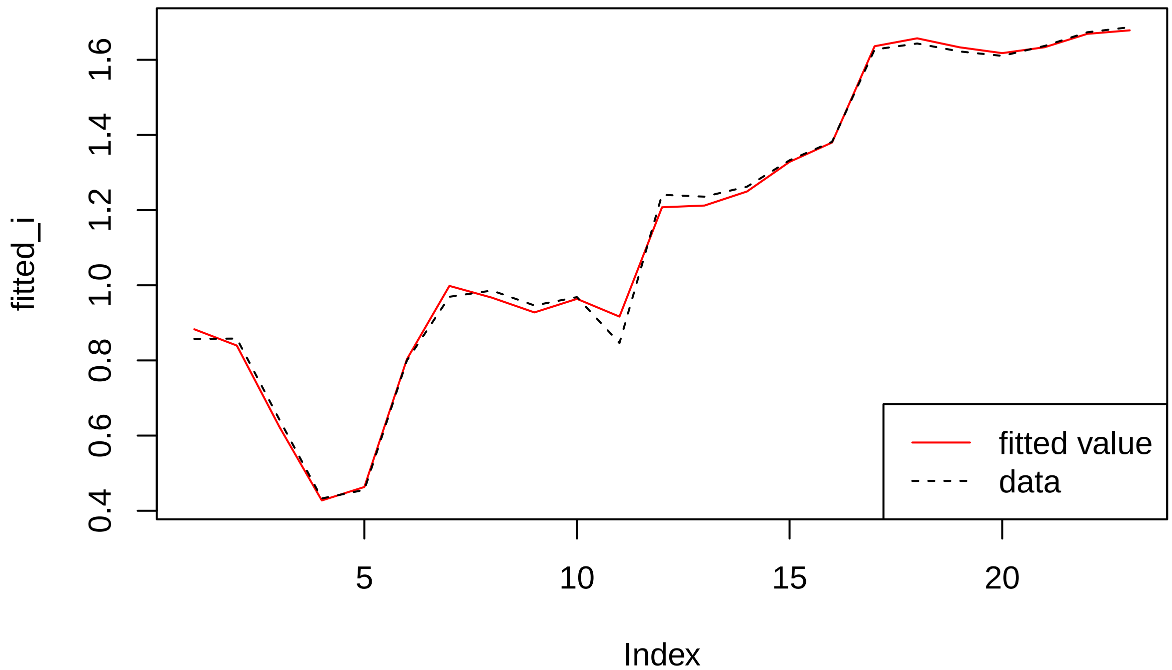


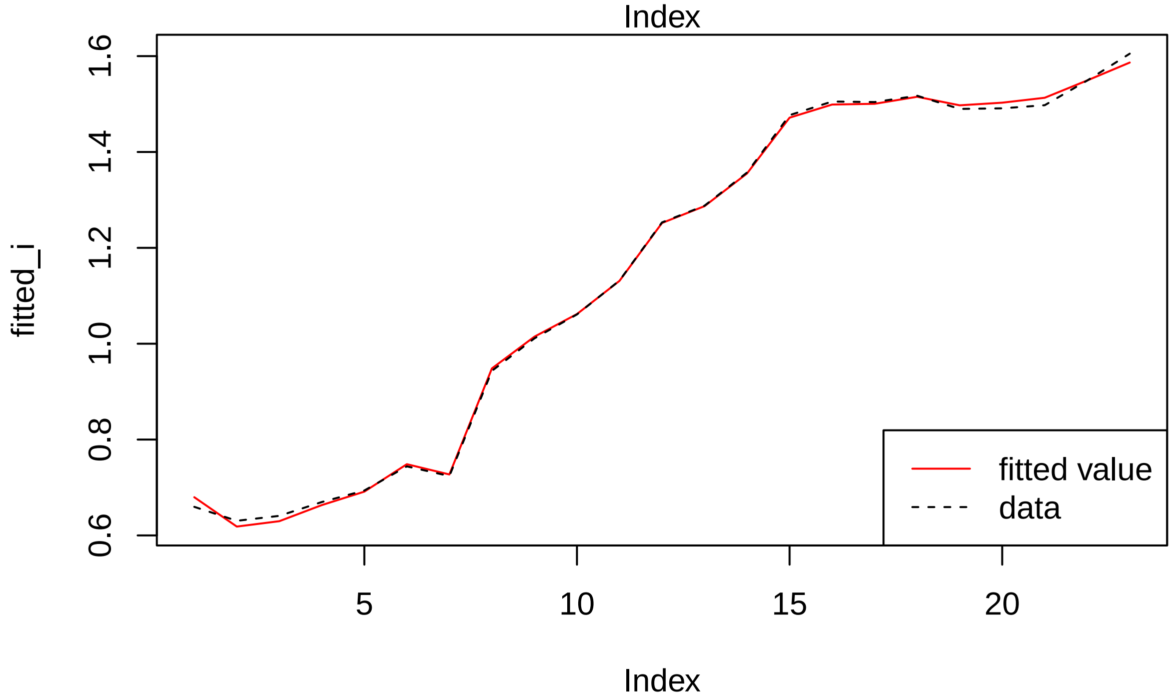


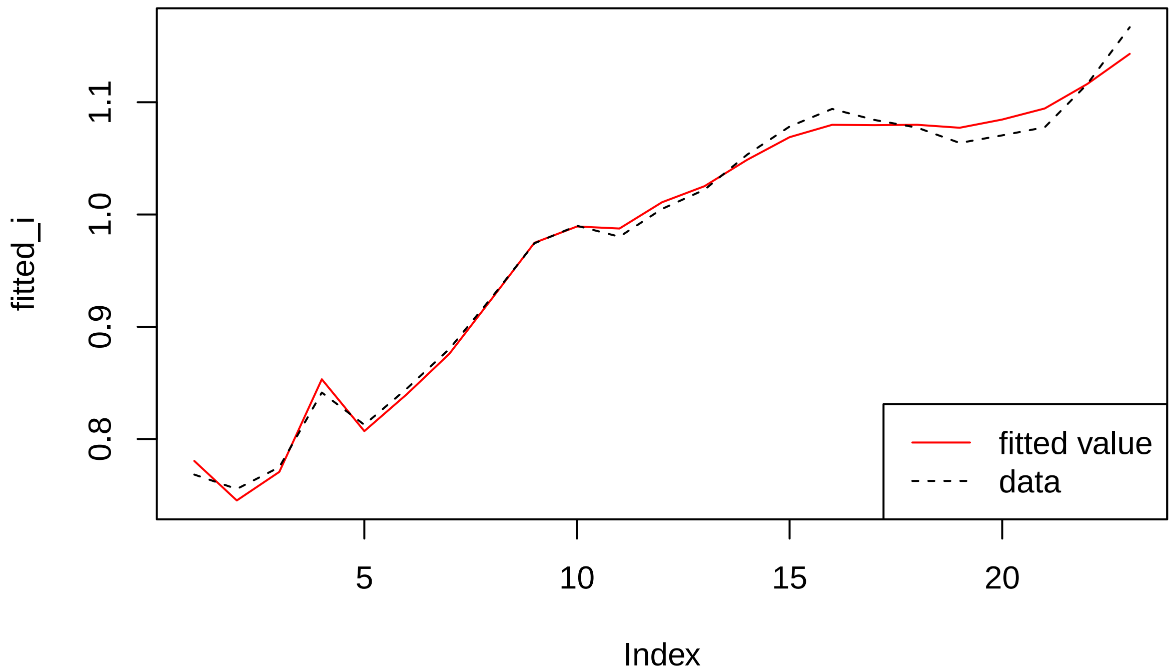
1.05
0.95
0.85
0.75
fitted_i
5
10
15
20
Index
fitted value
data

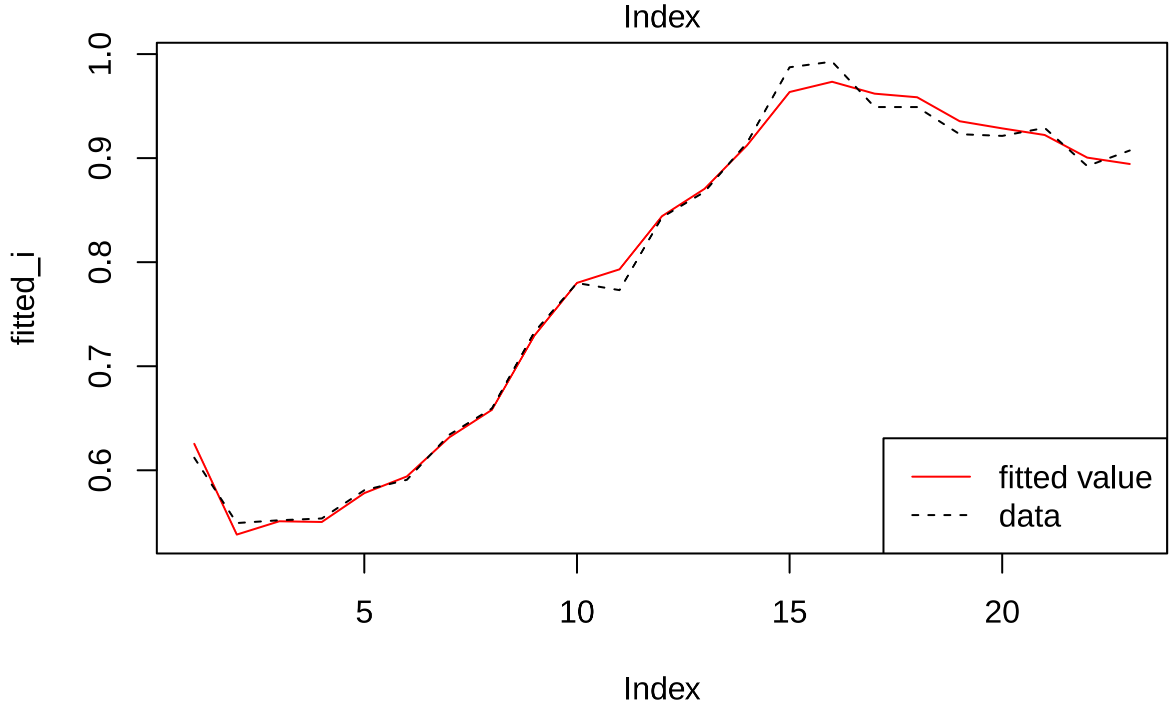


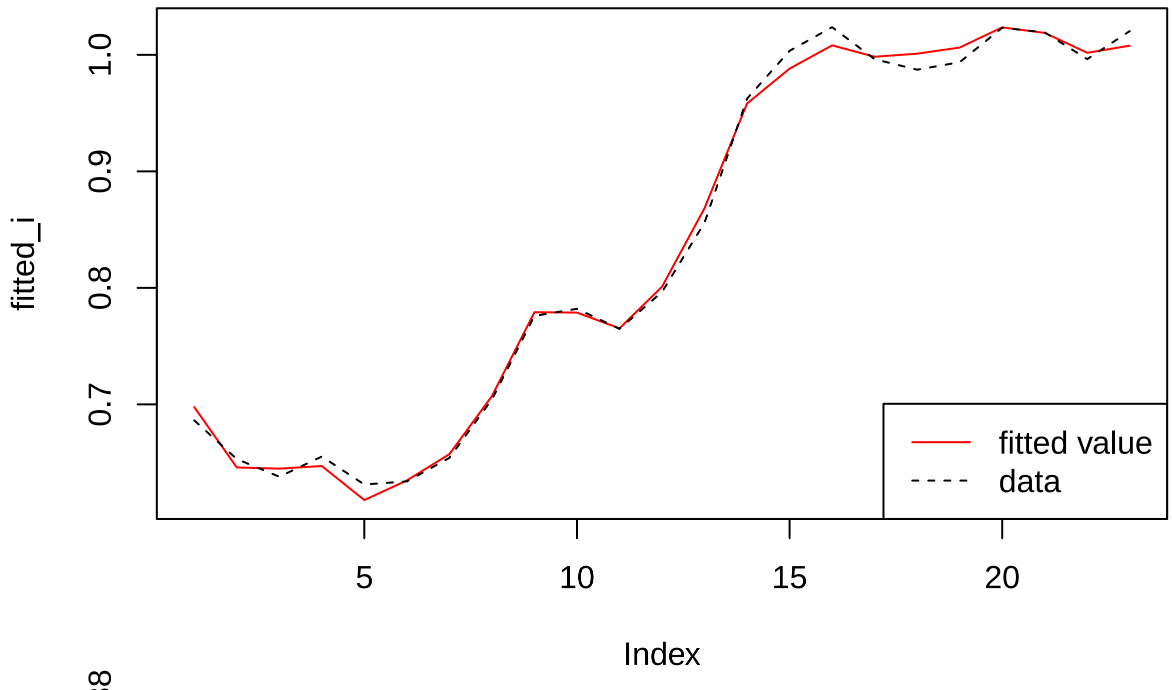
0.9
0.8
0.7
0.6
fitted_i
5
10
15
20
Index
fitted value
data


fitted_i
0.4
0.6
0.8
1.0
1.2
1.4
1.6
5
10
15
20
Index
fitted value
data

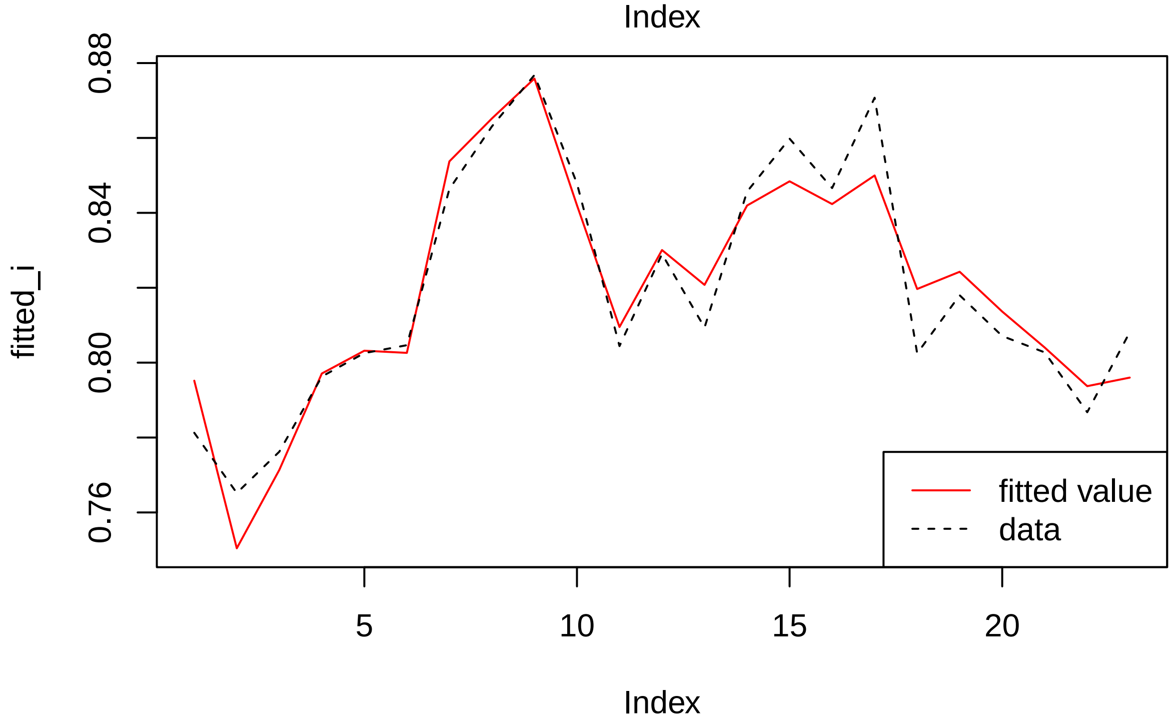

fitted_i
0.6
0.8
1.0
1.2
1.4
1.6
5
10
15
20
Index
fitted value
data


fitted_i
0.8
0.9
1.0
1.1
5
10
15
20
Index
fitted value
data

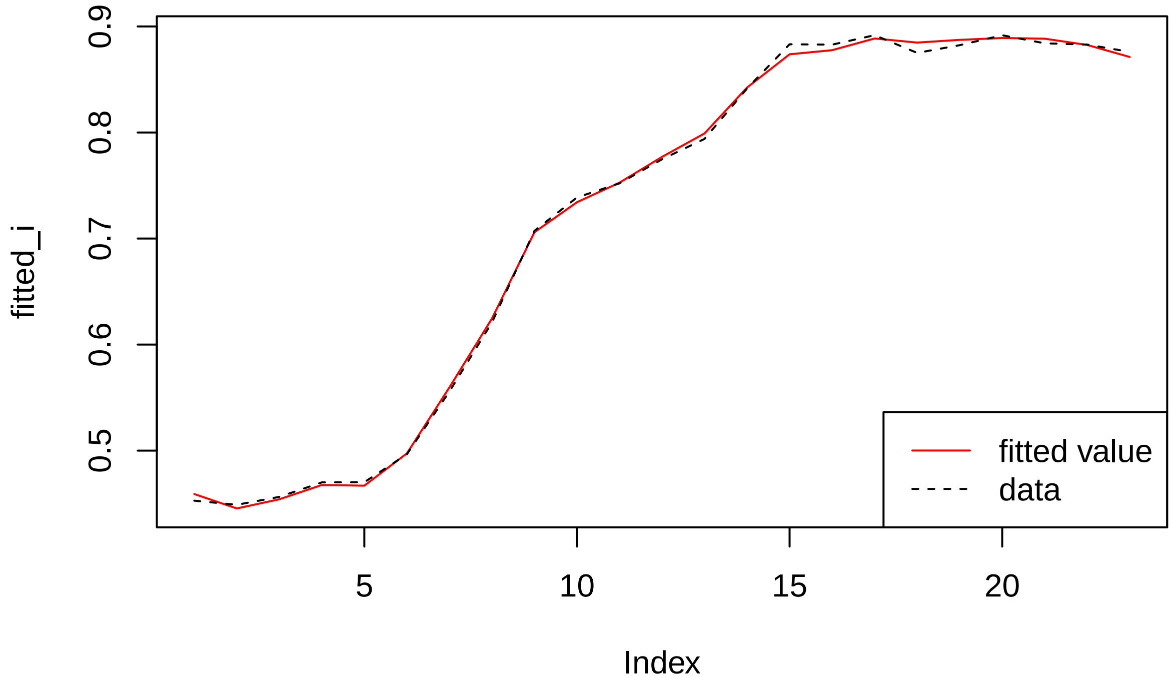

fitted_i
0.6
0.7
0.8
0.9
1.0
5
10
15
20
Index
fitted value
data



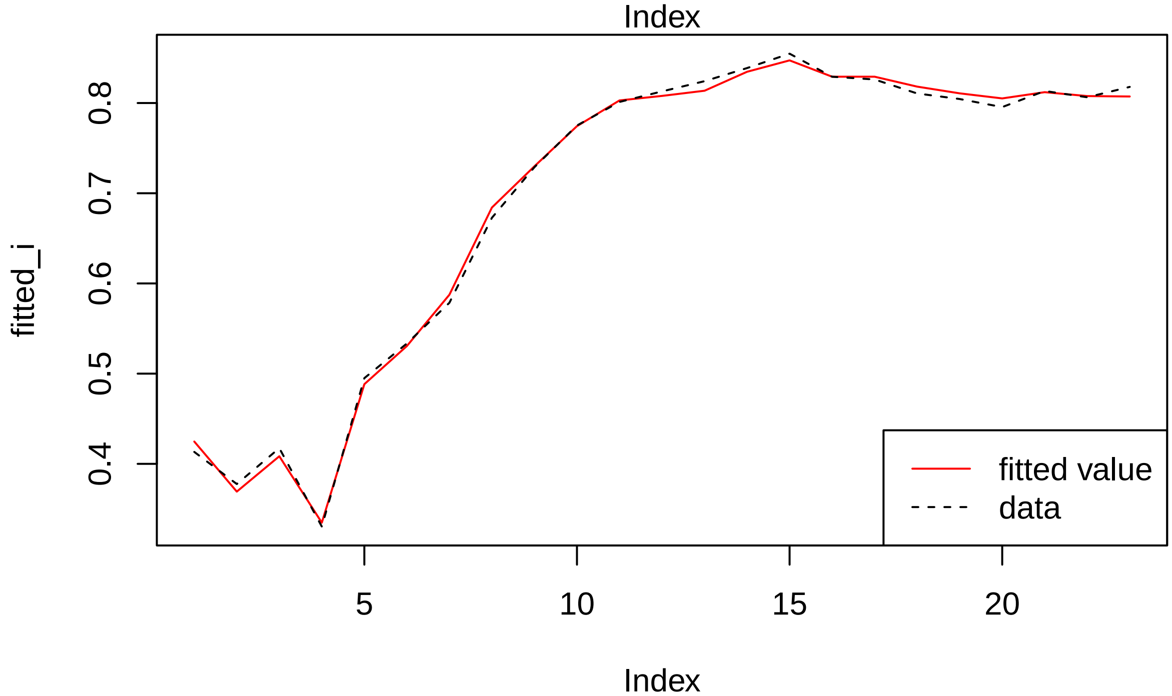
fitted_i
1.0
0.9
0.8
0.7
fitted value
data
5
10
15
20
Index



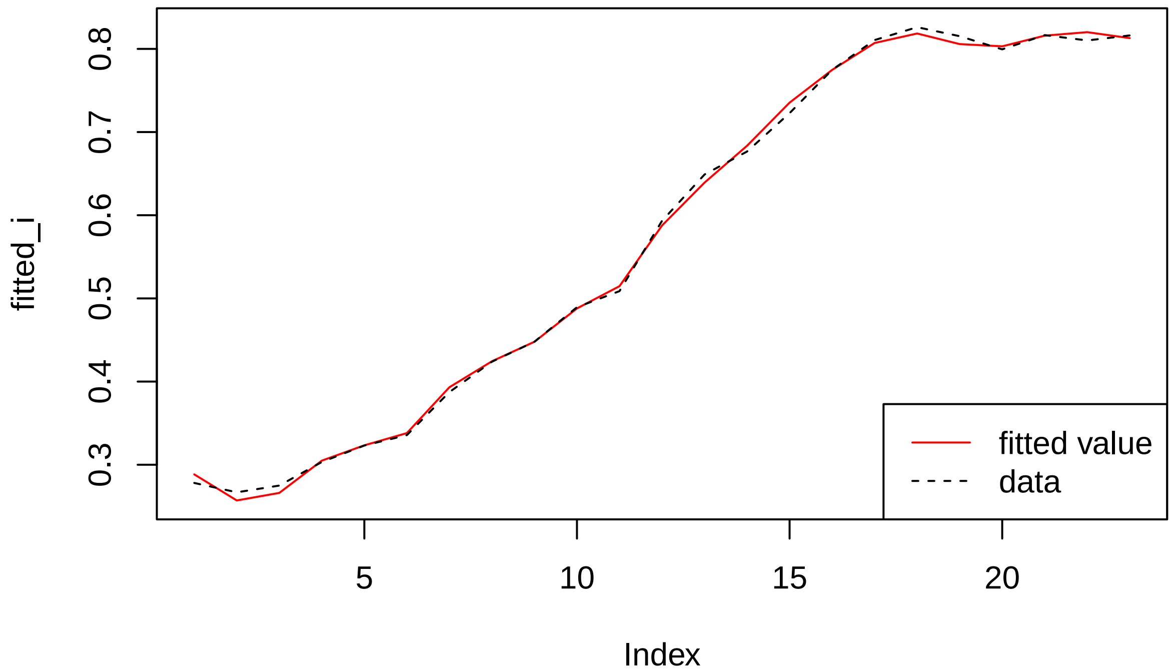
fitted_i
0.88
0.84
0.80
0.76
fitted value
data
5
10
15
20
Index

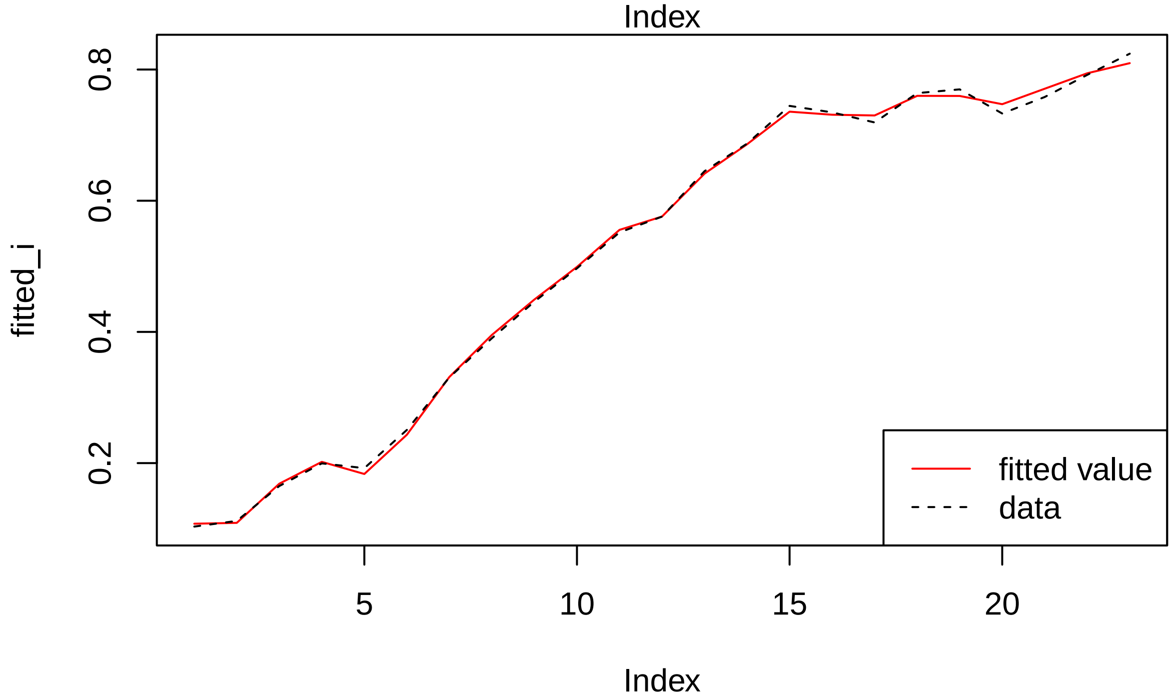


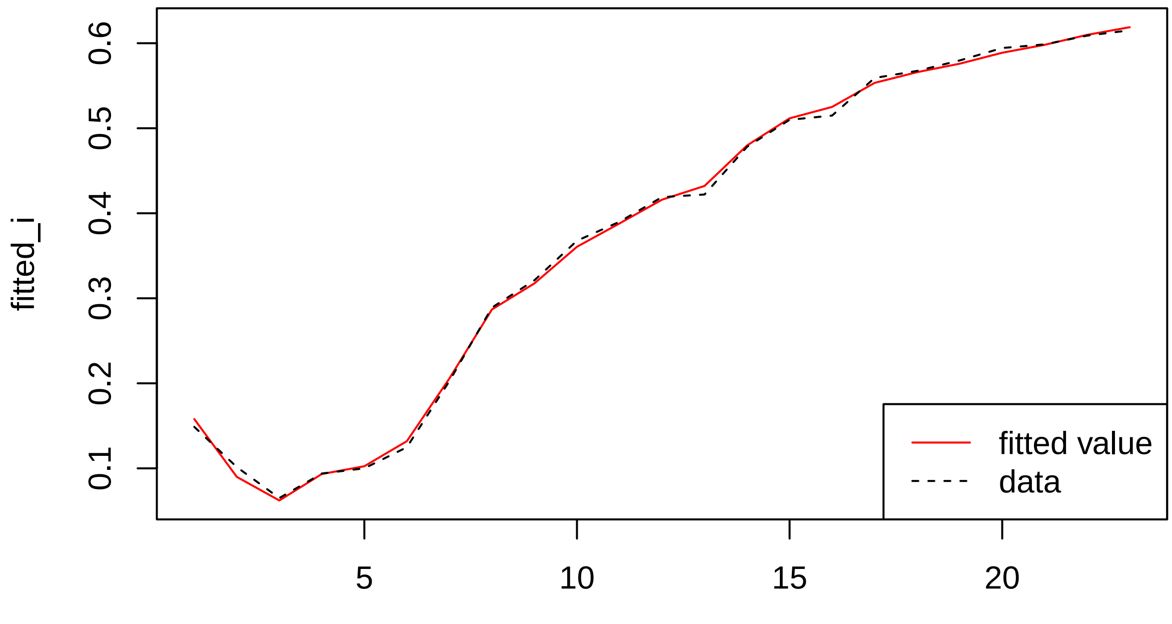
fitted_i
0.9
0.8
0.7
0.6
0.5
5
10
15
20
Index
fitted value
data



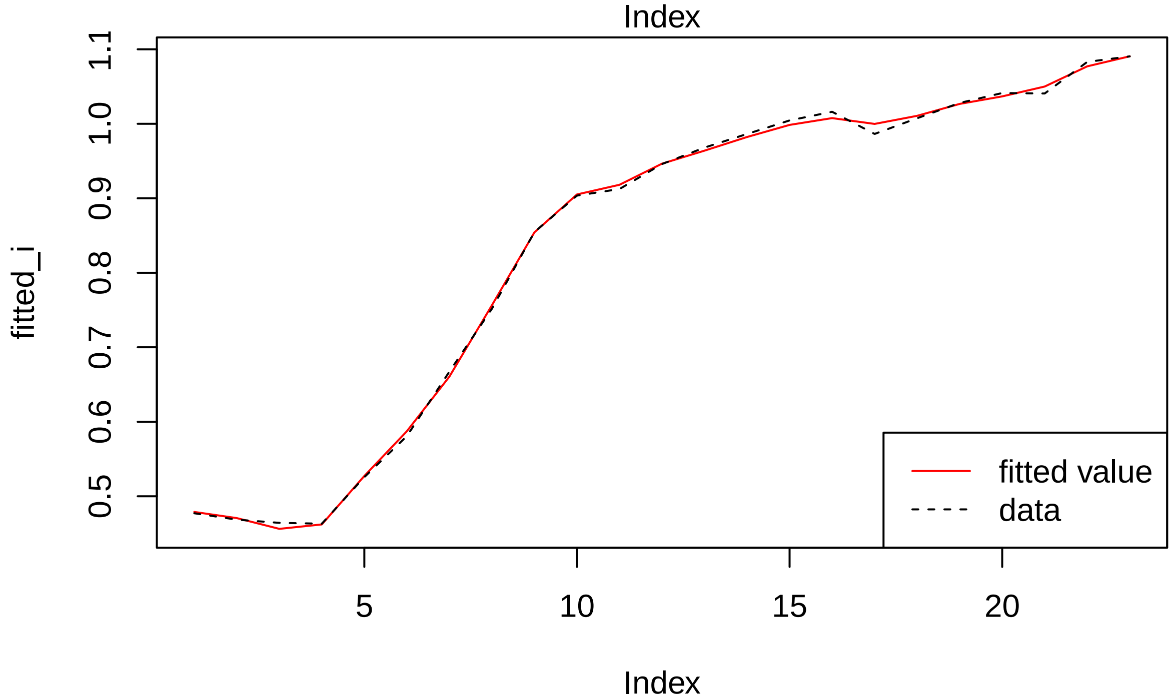
fitted_i
0.8
0.7
0.6
0.5
0.4
5
10
15
20
Index
fitted value
data


fitted_i
0.3
0.4
0.5
0.6
0.7
0.8
5
10
15
20
Index
fitted value
data

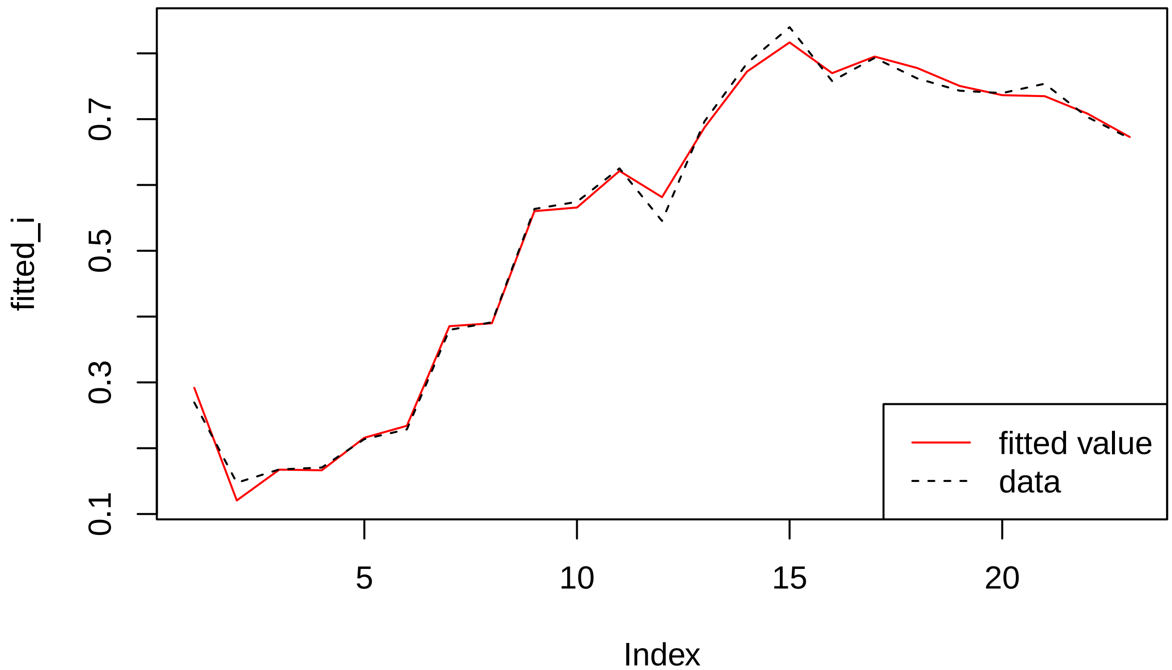

fitted_i
0.2
0.4
0.6
0.8
5
10
15
20
Index
fitted value
data

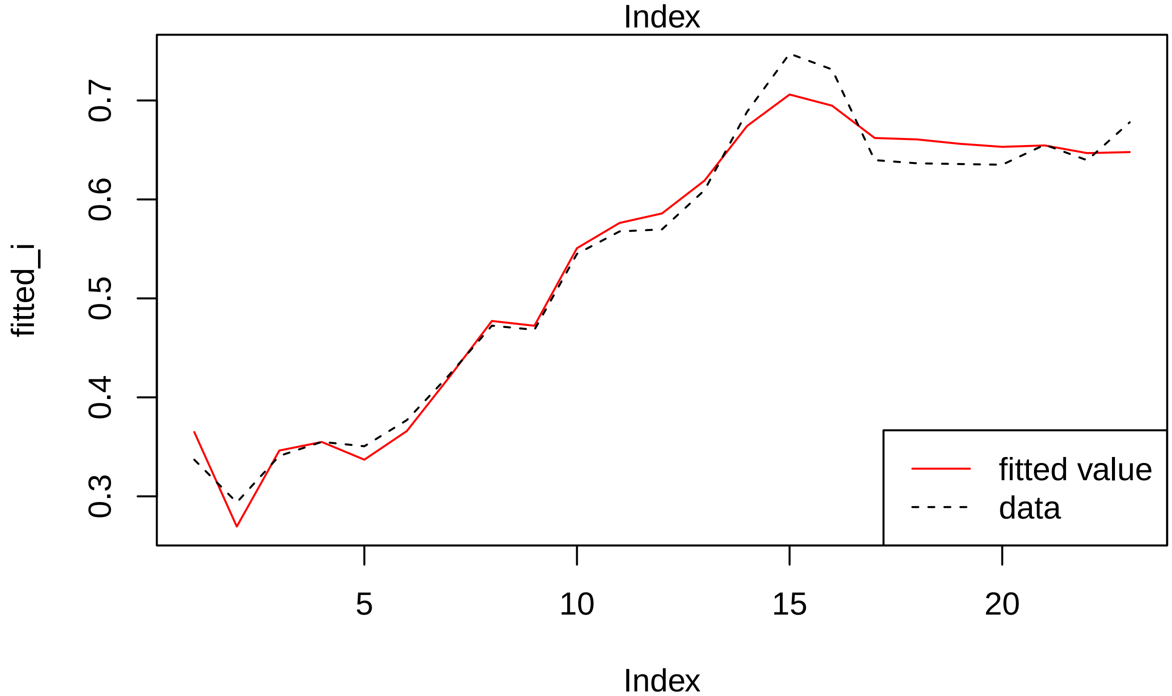


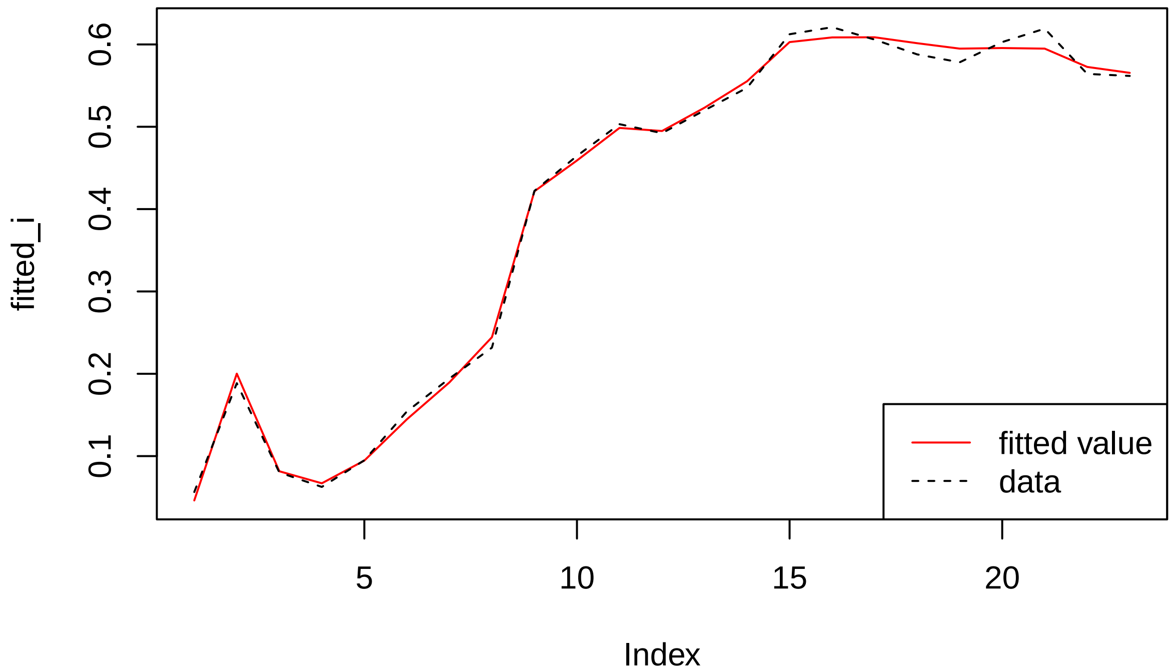
fitted_i
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
20
Index
fitted value
data



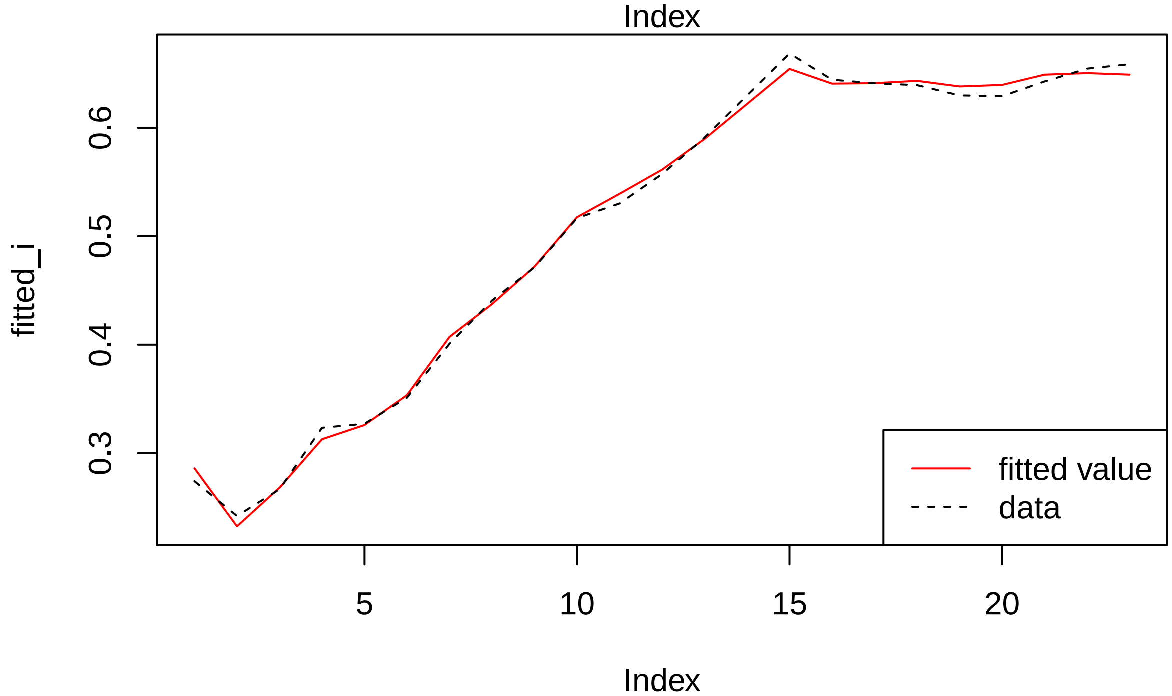
fitted_i
0.5
0.6
0.7
0.8
0.9
1.0
1.1
5
10
15
20
Index
fitted value
data


fitted_i
0.1
0.3
0.5
0.7
5
10
15
20
Index
fitted value
data

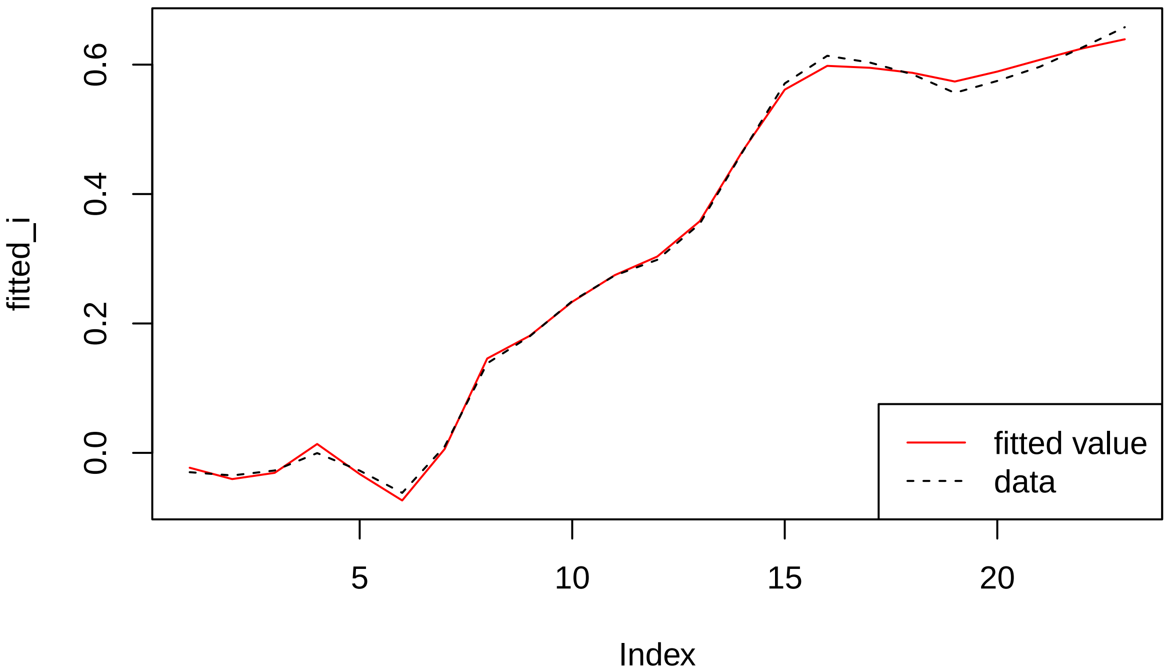

fitted_i
0.3
0.4
0.5
0.6
0.7
5
10
15
20
Index
fitted value
data

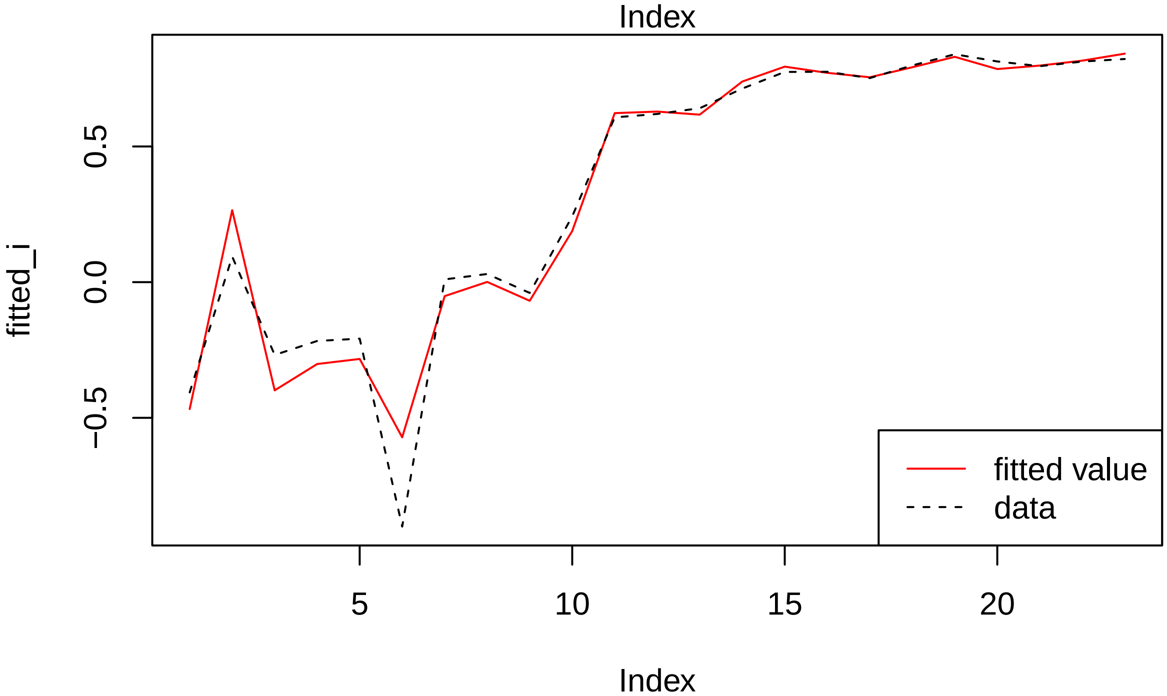


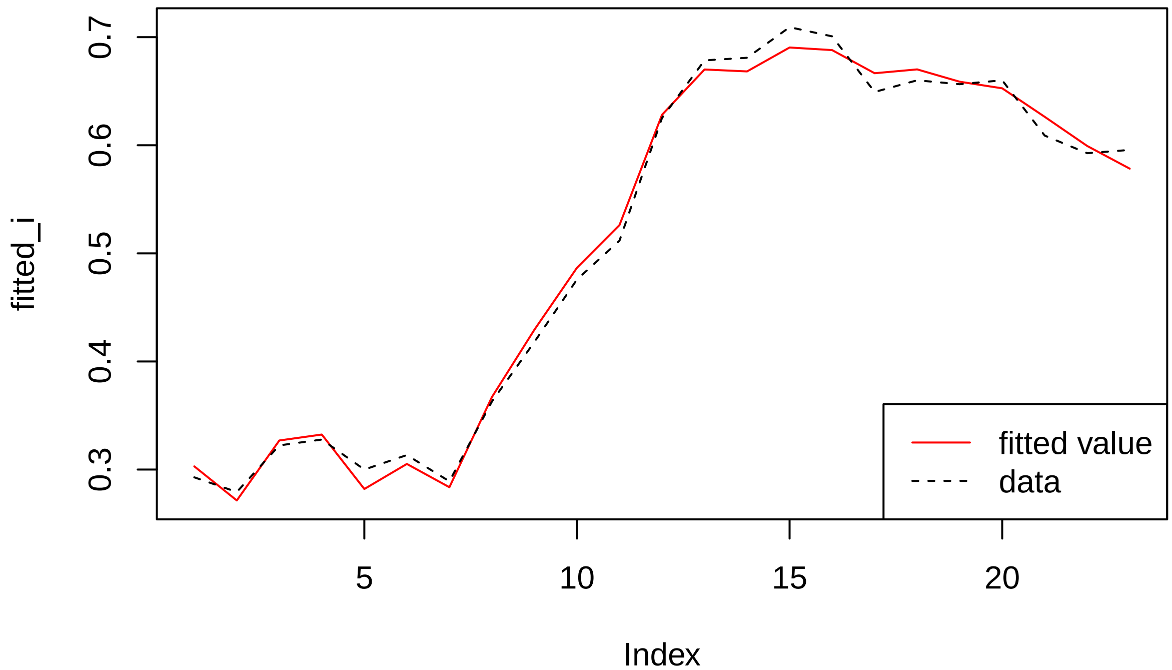
fitted_i
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
20
Index
fitted value
data



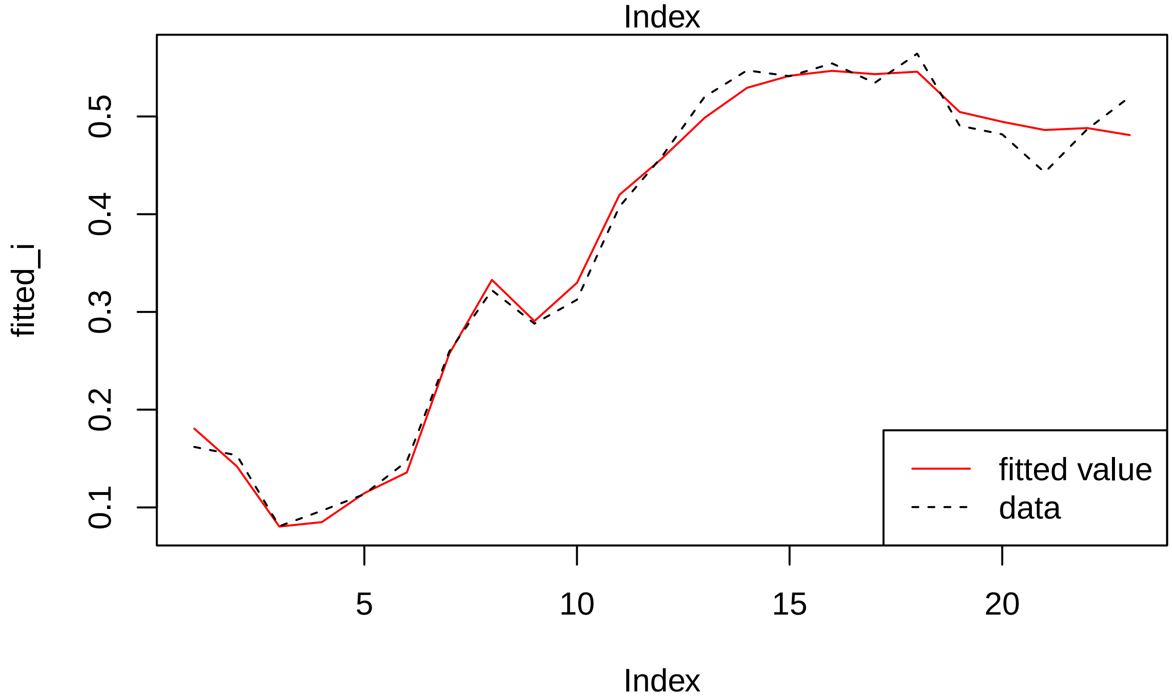
fitted_i
0.3
0.4
0.5
0.6
5
10
15
20
Index
fitted value
data


fitted_i
0.0
0.2
0.4
0.6
5
10
15
20
Index
fitted value
data

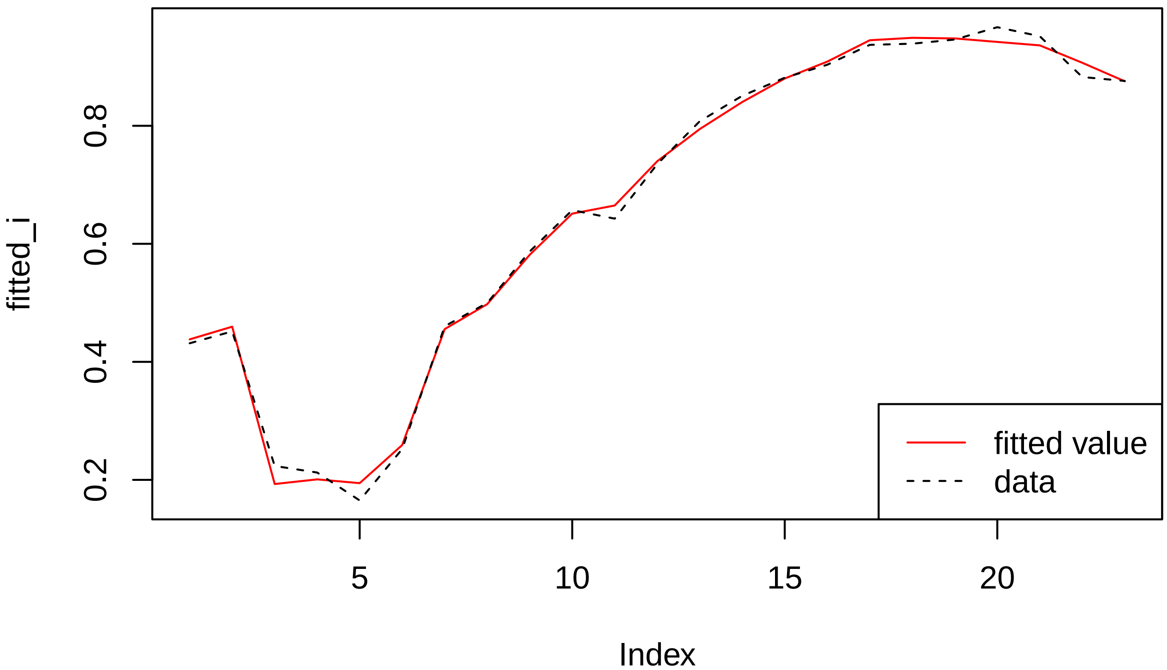

fitted_i
−0.5
0.0
0.5
5
10
15
20
Index
fitted value
data

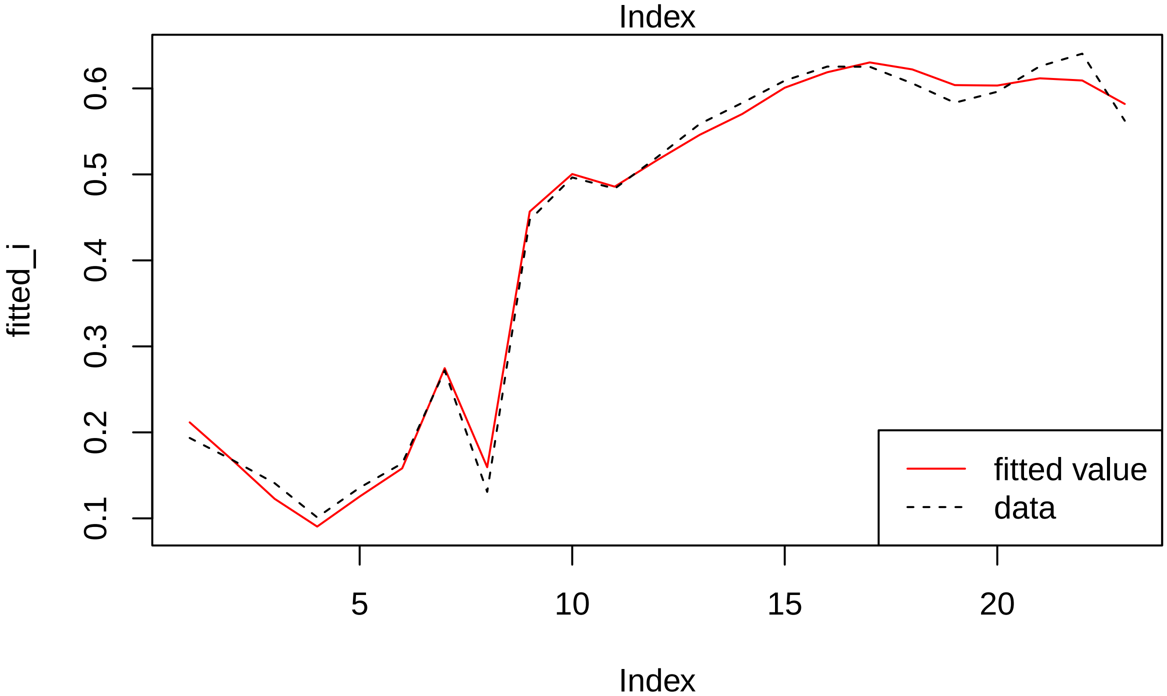


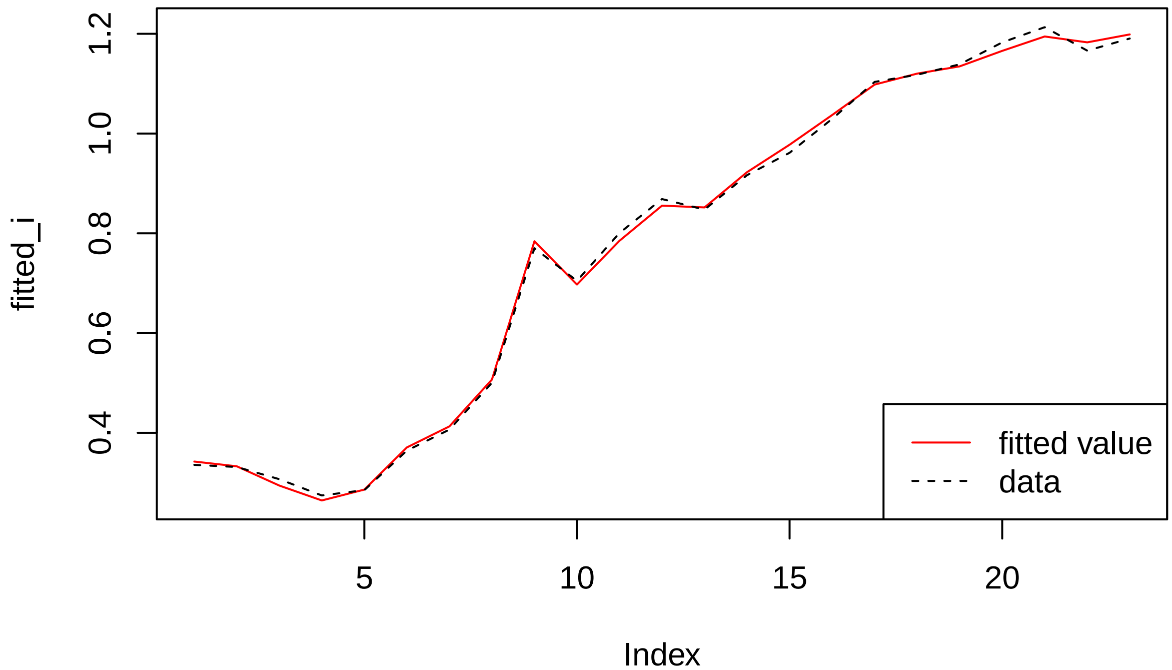
fitted_i
0.3
0.4
0.5
0.6
0.7
5
10
15
20
Index
fitted value
data

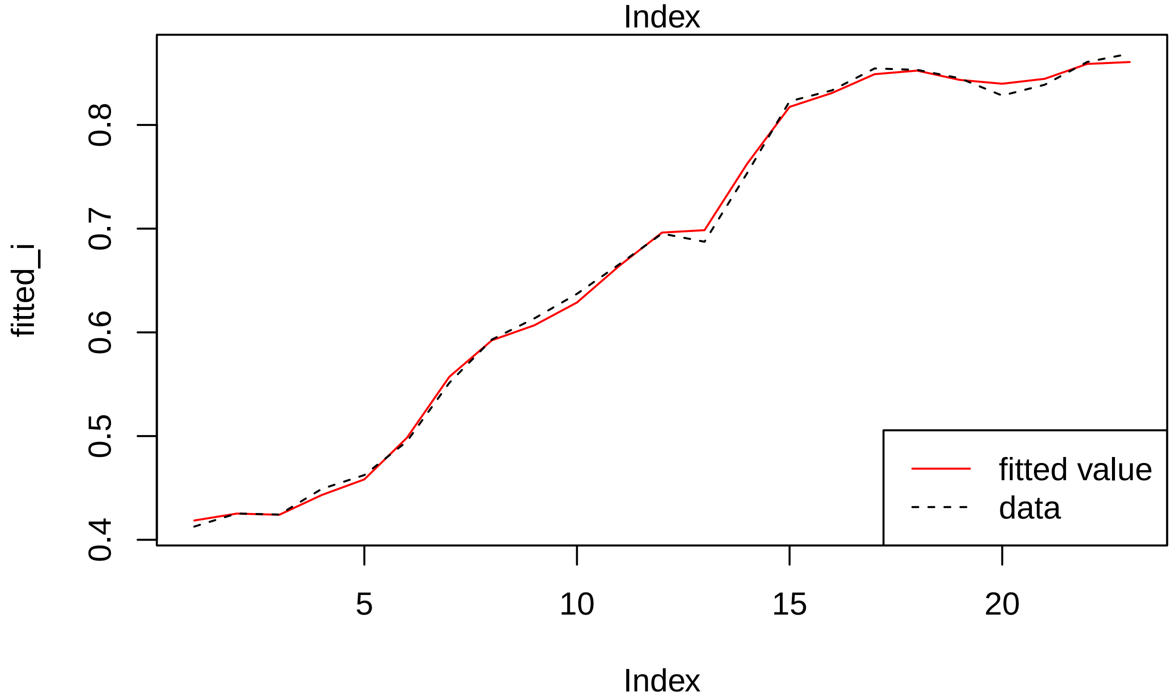


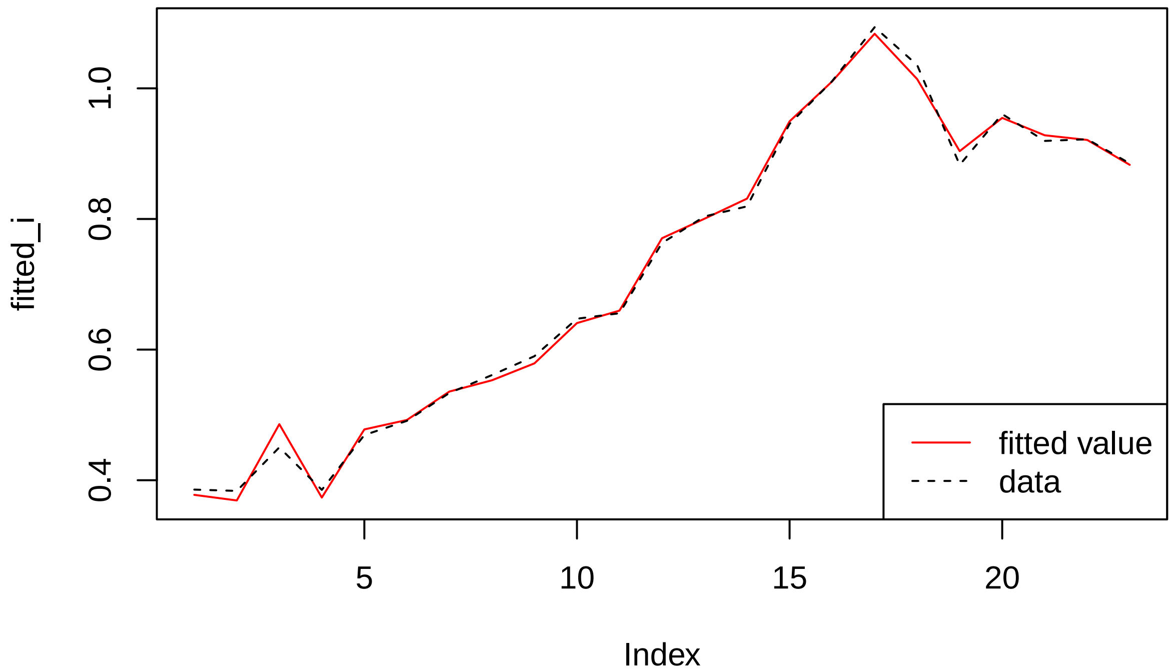
fitted_i
0.1
0.2
0.3
0.4
0.5
5
10
15
20
Index
fitted value
data

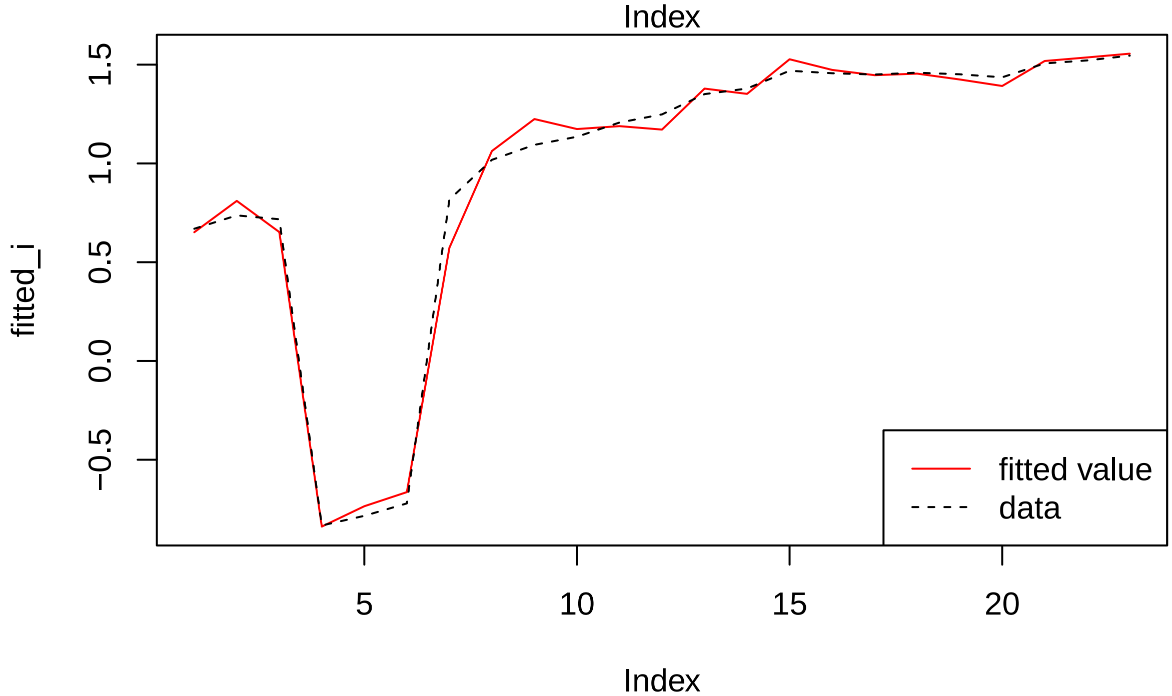

fitted_i
0.2
0.4
0.6
0.8
5
10
15
20
Index
fitted value
data

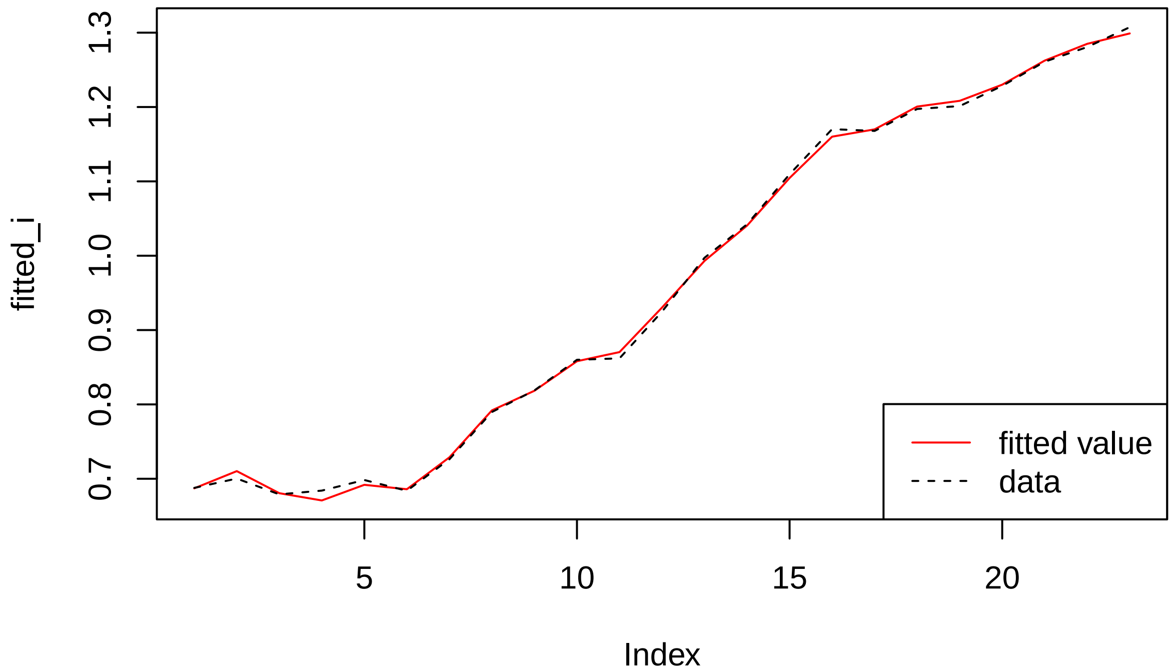

fitted_i
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
20
Index
fitted value
data


fitted_i
1.2
1.0
0.8
0.6
0.4
fitted value
data
5
10
15
20
Index


fitted_i
0.8
0.7
0.6
0.5
0.4
fitted value
data
5
10
15
20
Index


fitted_i
0.4
0.6
0.8
1.0
5
10
15
20
Index
fitted value
data


fitted_i
−0.5
0.0
0.5
1.0
1.5
5
10
15
20
Index
fitted value
data




```
coefficient
```

```
##             [,1]       [,2]         [,3]          [,4]        [,5]        [,6]
##  [1,] -4.8189682  4.6742841 -1.044681888  1.473127e-03 0.052310577  0.24815465
##  [2,] -2.5593877  2.2968694 -0.293261504  2.012333e-03 0.025200624 -0.39678111
##  [3,] -1.2559808  1.1365204 -0.084080076  9.649689e-04 0.018176320  0.53167700
##  [4,] -0.6820402  0.4149705  0.162993089  4.646629e-04 0.010181819  0.85394757
##  [5,] -0.7548468  0.7038291  0.052567598  6.696459e-06 0.012198243  0.63760784
##  [6,] -2.5263428  3.0229209 -0.624528155  4.187748e-04 0.015548410 -0.40568993
##  [7,] -1.4526408  1.7768576 -0.273666505  1.614775e-03 0.012220522 -0.03133990
##  [8,] -1.0911174  0.6059923  0.016664013  5.292824e-04 0.019128595  1.48824753
##  [9,] -4.8171121  4.3641857 -0.881658691  3.350570e-04 0.056397504  0.14492937
## [10,] -1.7665800  1.5464984 -0.194238357  9.296834e-05 0.034323903  0.28147806
## [11,] -2.7763701  2.7353373 -0.446423691  3.690379e-03 0.035014957 -0.66275140
## [12,] -1.1525419  0.8269376  0.021841161 -1.471803e-04 0.024153679  0.64851398
## [13,] -2.3458125  1.9347860 -0.226713643  9.429720e-04 0.060965570 -0.36910291
## [14,] -1.4537566  1.4804588 -0.268665957  9.139022e-04 0.026034248  0.06469179
## [15,] -2.0437579  2.6702695 -0.537421326 -2.858770e-03 0.018130237 -0.71965153
## [16,] -0.9704031  0.4018767  0.229519265  1.511686e-03 0.024845216  0.74969625
## [17,] -2.3128950  2.5245266 -0.517065656  9.957481e-04 0.029869732 -0.20959481
## [18,] -1.8350560  1.9381057 -0.307636953  3.479888e-03 0.028456942 -0.32602624
## [19,] -1.6320796  1.2880701 -0.157888929 -6.919962e-03 0.059067948  0.34367223
## [20,] -1.5061266  0.9532385  0.006889319 -6.179298e-04 0.045873026  0.38824800
## [21,] -4.9644992  5.9427628 -1.314143541 -1.447742e-02 0.077326260 -3.02611019
## [22,] -1.8487197  1.5451184 -0.219665230 -3.866540e-03 0.036435453  0.80506158
## [23,] -1.9727170  2.1059168 -0.334099758  5.420263e-03 0.029766542 -0.53631454
## [24,] -0.8680848  1.0699511  0.022335124  6.163595e-04 0.008879797  0.35922238
## [25,] -1.5712022  1.5265911 -0.190314473 -6.714400e-04 0.027083642  0.20709583
## [26,] -0.9176701  0.7481672  0.011060464 -2.564639e-04 0.017454851  0.65770184
## [27,] -1.1528633  1.4177735 -0.205574274 -1.652022e-03 0.014859263  0.12311777
## [28,] -1.1297873  0.3849442  0.254887884  1.705735e-03 0.020911164  1.16592473
## [29,] -0.8930363 -0.8690210  0.788082052 -6.040132e-04 0.023319282  1.24064830
```

```
## [30,] -0.3664528  0.3628599  0.092003766 -6.454178e-04 0.013143193  0.71054641
##               [,7]
##  [1,] -0.109792486
##  [2,] -0.014357698
##  [3,] -0.496357490
##  [4,] -0.496578574
##  [5,] -0.410916561
##  [6,]  0.289953360
##  [7,] -0.209563454
##  [8,] -0.882181284
##  [9,] -0.034809208
## [10,] -0.347440378
## [11,]  0.174631917
## [12,] -0.664267563
## [13,]  0.039235774
## [14,] -0.003918757
## [15,]  0.482071631
## [16,] -1.049741122
## [17,]  0.024772108
## [18,] -0.092783992
## [19,] -0.294768260
## [20,] -0.373666954
## [21,]  1.880048955
## [22,] -0.808358656
## [23,] -0.059019419
## [24,] -0.907070917
## [25,] -0.465947803
## [26,] -0.390492054
## [27,] -0.178047907
## [28,] -1.190695441
## [29,] -1.052208466
## [30,] -0.392956361
```