REM Tutorial 4: Regression II - Panel – Workbook

Prof. Dr. Kathleen Kürschner Rauck

AT 2023

Contents

Practical 1: Data Preparation and Exploration **2**
a. Load the **Bulwien_Gesa** and **Bulwien_Gesa_Geodat** data into R Studio and merge the two data sets. . . . 2
b. Plot the city-locations on a map of Germany using the `map()` function and using the city names as markers. . . . 3
c. Using circle markers, add information on cities' average economic activity to the map. . 5

Practical 2: Regression Analysis **7**
a. Run simple OLS regressions, with and without variable transformations, to enquire into the relationship between office rents and employment. . . . 7
b. Use a log-log specification to model the relationship between office rents and employment. Include a full set of dummy variables (DVs) for time periods as well as observation entities in your model. . . . 9
c. Replicate your result from part b. using the `plm` package. . . . 11
d. Add further regressors to the model. . . . 13

Remember to **organise your workspace** (create a working folder and store the **Bulwien_Gesa** and **Bulwien_Gesa_Geodat** data I uploaded on Canvas), **set up a new R Script**, **set your working directory**, and save the R Script in your working directory.

For this practical, we need to install several packages: `maps`, `mapdata`, `bdsmatrix`, `collapse`, `sandwich`, `lmtest`, `maxLik` (and dependencies: `miscTools`, `generics`), `Rdpack` (and dependency: `rbibutils`), `Formula`, `plm`

Now we can solve some practical problems provided in exercise sheet *Tutorial 4 - Problem Set*.

Practical 1: Data Preparation and Exploration

The data comes in two files, which is commonly the case in practice (i.e., we often get several files when dealing with large real estate data).

a. Load the Bulwien_Gesa and Bulwien_Gesa_Geodat data into R Studio and merge the two data sets.

Here we have one data set (**Bulwien_Gesa**) that contains information on average office rents (*rent*), office stock (*stock*), and vacancy rates (v) for several (42) German cities. It also contains information on two city-level socio-economic characteristics, i.e., employment (*emp*) and gross value added (*gva*). We load this dataset and store it in R object **dat1**.

The second data set contains spatial information, i.e., geo-codes (*latitude* and *longitude*), for the 42 cities' centroids. We load this dataset and store it in R object **dat1.geo**.

```
# load data
dat1 <- read.table("Bulwien_Gesa.txt", header = TRUE)
dat1.geo <- read.table("Bulwien_Gesa_Geodat.txt", header = TRUE)
```

Now we can combine (merge) the two datasets into a single data frame, called **dat2**. Before we apply the `merge()` function, we enquire into the dimensions of the two data frames **dat1** and **dat1.geo**, respectively. Type:

```
dim(dat1)
```

```
## [1] 798   7
```

```
dim(dat1.geo)
```

```
## [1] 42  3
```

The data sets are of different size because:

i. the geo-locations of the city centroids are time-invariant ⇒ there is only one observation (row) for each of the 42 cities in **dat1.geo**.

ii. office rents etc. are not constant over time ⇒ for each city, there are 19 annual observations on these time-varying characteristics, comprising the years 1990-2008 (i.e., $19 \times 42 = 798$ rows in **dat1**).

To merge the two data sets, we need to identify a variable, that is common to both, based on which the merge should be executed. Here, both data frames contain variable *City*, whose attributes are the names of the 42 cities (character strings) with identical spelling. To check this, type:

```
levels(as.factor(dat1$City))
```

```
##  [1] "Berlin"              "Bochum"              "Bonn"
##  [4] "Bremen"              "Chemnitz"            "Cottbus"
##  [7] "Dortmund"            "Dresden"             "Duisburg"
## [10] "Duesseldorf"         "Frankfurt.(Main)"    "Freiburg.(Breisgau)"
## [13] "Halle.(Saale)"       "Hamburg"             "Hannover"
## [16] "Karlsruhe"           "Kassel"              "Kiel"
## [19] "Koeln"               "Leipzig"             "Leverkusen"
## [22] "Luebeck"             "Magdeburg"           "Mainz"
## [25] "Muenchen"            "Muenster"            "Neuss.(Stadt)"
## [28] "Nuernberg"           "Oberhausen"          "Potsdam"
## [31] "Regensburg"          "Rostock"             "Saarbruecken.(Stadt)"
## [34] "Schwerin"            "Stuttgart"           "Ulm"
## [37] "Wiesbaden"           "Wilhelmshaven"       "Wolfsburg"
## [40] "Wuppertal"           "Wuerzburg"           "Zwickau"
```

```
levels(as.factor(dat1.geo$City))
```

```
##  [1] "Berlin"              "Bochum"              "Bonn"
##  [4] "Bremen"              "Chemnitz"            "Cottbus"
##  [7] "Dortmund"            "Dresden"             "Duisburg"
## [10] "Duesseldorf"         "Frankfurt.(Main)"    "Freiburg.(Breisgau)"
## [13] "Halle.(Saale)"       "Hamburg"             "Hannover"
## [16] "Karlsruhe"           "Kassel"              "Kiel"
## [19] "Koeln"               "Leipzig"             "Leverkusen"
## [22] "Luebeck"             "Magdeburg"           "Mainz"
## [25] "Muenchen"            "Muenster"            "Neuss.(Stadt)"
## [28] "Nuernberg"           "Oberhausen"          "Potsdam"
## [31] "Regensburg"          "Rostock"             "Saarbruecken.(Stadt)"
## [34] "Schwerin"            "Stuttgart"           "Ulm"
## [37] "Wiesbaden"           "Wilhelmshaven"       "Wolfsburg"
## [40] "Wuppertal"           "Wuerzburg"           "Zwickau"
```

Hence, we can use this variable to merge the variables that are contained in **dat1.geo** but not in **dat1** (i.e., the geo-codes) to **dat1**, based on exact matches in the attributes of variable *City*, which is contained in both data frames. To do this, we specify the option `by = "City"` when we apply the `merge()` function. Type:

```
dat2 <- merge(dat1, dat1.geo, by = "City")
View(dat2)
```

The merge was successful, i.e., all variables are combined in one data frame (**dat2**), which we will henceforth use.

b. Plot the city-locations on a map of Germany using the `map()` function and using the city names as markers.

This time, we introduce a different package/function to plot data on a map. We start by loading the packages `maps` and `mapdata`. The former loads maps, the latter contains a map of Germany in low- and high-resolution.

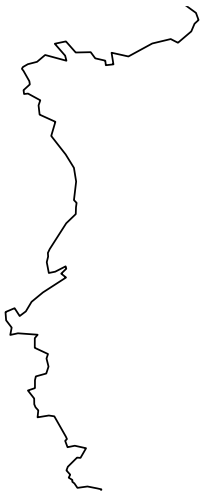
```
library(maps)
library(mapdata)
```

Now we can plot a simple map that only displays the border of Germany. The `map()` function allows to plot such simple lines or polygons of any shape, accessing a database, which contains a list of y & x (e.g., latitude & longitude) coordinates for all the individual points that are required to draw a certain line/polygon. The database we specify is called *worldHires*, which contains the coordinates of the borders of the countries around the globe (*world*) in high resolution (*Hires*).

```
map("worldHires",        # accesses coordinates in database "worldHires"
    regions = "Germany") # only selects the polygon, named "Germany" from database
```

Alternatively, you can tweak the map to look at only a certain sub-region of Germany. To plot, for instance, the Rhine-Ruhr metropolitan region, you can add the options `ylim` and `xlim` to the `map()` command in order to specify limits for latitude (corresponding to the values on the y-axis) and longitude (corresponding to the values on the x-axis), respectively.

```
map("worldHires",regions = "Germany",
    ylim = c(50.5,52), xlim = c(5,8.5)) # only displays the Rhine-Rhur region
```



Next, we can plot the locations of the 42 city centroids onto the map, using the geo-codes contained in data frame **dat2**. Note that we plot the location of each city-centroid only once by means of the `unique()` function, which ignores duplicate observations in specified variables.

Furthermore, instead of using, e.g., point markers, we would like to plot the actual names of the cities to mark their locations. This is why we apply the `text()` function, specifying the marker locations (given by variables *longitude* and *latitude* of **dat2**) and telling your PC where to find the corresponding labels/names to be printed at the specified locations.

```
# plot again base map for entire Germany
map("worldHires", regions = "Germany")

# new code
text(unique(dat2$longitude), unique(dat2$latitude), # x-y coordinates for labels
     labels = unique(dat2$City), # names/labels to plot at specified locations
     cex = 0.6) # marker/character size (numeric Character EXpansion factor)
```

c. Using circle markers, add information on cities' average economic activity to the map.

The size of the circle markers could be a function of a city's gross value added (*gva*). Therefore, we first generate an R object, called **mean.gva**, in which we store the average value of variable *gva* for each of the 42 cities. To accomplish this, we can use the `tapply()` function, which applies/repeats a function (here: 'calculate the mean of') to an R object (here: 'variable *gva*'), which can be split into groups/subsets according to some indexing R object (here: 'by expressions of variable *City*'). General note: the indexing R object/variable must always be of same length as the R object to which the function is applied, such that each element is in fact indexed.

```
# calculate mean of gva by city
mean.gva <- tapply(dat2$gva,  # object to which function is applied
                   dat2$City, # index by which this object is split into groups
                   mean,      # the function to apply
                   na.rm = TRUE) # removes missing values
class(mean.gva)
```

```
## [1] "array"
```

```
dim(mean.gva)
```

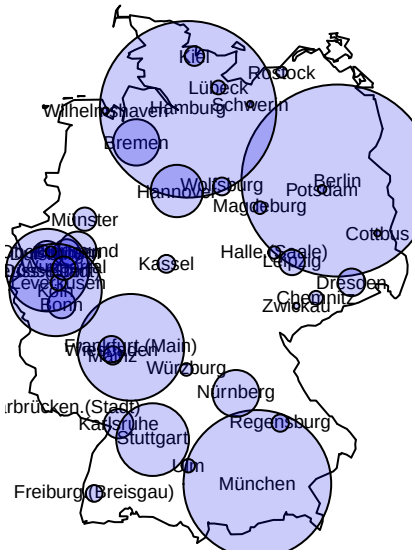
```
## [1] 42
```

The R object **mean.gva** contains the 42 mean values of gross value added (i.e., one for each city) in chronological order (i.e., Berlin, Bochum, etc.).

Next, we employ these values to add circle markers to the city locations using the `symbols()` function. We specify the option `circles = mean.gva` to tell our PC to use circles as the marker symbol and to draw these circles of different size, according to the values contained in array **mean.gva**, which, as we know, contains the mean gva values in the same city-order (i.e.,Berlin, Bochum, etc.) as the city-order of coordinates in R object **dat2**.

```
# reproduce map from before
map("worldHires", regions = "Germany")
text(unique(dat2$longitude), unique(dat2$latitude), labels=unique(dat2$City), cex=0.6)

# new code
symbols(unique(dat2$longitude), unique(dat2$latitude), # again city locations
        circles = mean.gva, # circles of specified sizes (radius)
        add = TRUE,         # adds to existing plot
        inches = 0.5, # scales size of circles (largest mean.gva is 0.5 inches)
        bg = rgb(0,0,0.9,0.2)) # fills circles with specified colour & opacity
```



Practical 2: Regression Analysis

a. Run simple OLS regressions, with and without variable transformations, to enquire into the relationship between office rents and employment.

We begin our analysis with a simple pooled OLS regression of variable *rent* on variable *emp* from dataset **dat2**, i.e., we can use again the `lm()` function.

Model 1: Simple pooled OLS without variable transformations

```
options(scipen = 5) # general setting to turn off scientific notation globally
                    # note: specify scipen = 0 to return to default setting

pooled.ols1 <- lm(rent ~ emp, data = dat2)
summary(pooled.ols1)
```

```
##
## Call:
## lm(formula = rent ~ emp, data = dat2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6972 -1.4658 -0.3583  0.9663 12.3711
##
## Coefficients:
##                 Estimate   Std. Error t value Pr(>|t|)
## (Intercept) 7.4901518020 0.1150864209   65.08   <2e-16 ***
## emp         0.0000248113 0.0000007271   34.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.283 on 690 degrees of freedom
##   (106 observations deleted due to missingness)
## Multiple R-squared:  0.6279, Adjusted R-squared:  0.6274
## F-statistic:  1164 on 1 and 690 DF,  p-value: < 2.2e-16
```

Without any variable transformation the estimated coefficient on *emp* is very small. This is not surprising, since rents are measured in terms of rent per square metre and employment is measured in terms of employed persons:

```
summary(dat2[,5:6])
```

```
##       rent             emp
##  Min.   : 5.11   Min.   : 14226
##  1st Qu.: 7.67   1st Qu.: 34903
##  Median : 9.20   Median : 56510
##  Mean   :10.28   Mean   :103932
##  3rd Qu.:11.76   3rd Qu.:102859
##  Max.   :28.63   Max.   :590605
##                  NA's   :106
```

and a single additional person is likely not to exert much of an effect on office rent per square metre. Alternatively, we can transform variable *emp* in units of thousands and re-run the regression.

Model 2: Simple pooled OLS with transformed employment in units of thousands

```
pooled.ols2 <- lm(rent ~ I(emp/1000), data = dat2) # employment in units of 1k
summary(pooled.ols2)
```

```
## Call:
## lm(formula = rent ~ I(emp/1000), data = dat2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6972 -1.4658 -0.3583  0.9663 12.3711
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.4901518  0.1150864   65.08   <2e-16 ***
## I(emp/1000) 0.0248113  0.0007271   34.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.283 on 690 degrees of freedom
##   (106 observations deleted due to missingness)
## Multiple R-squared:  0.6279, Adjusted R-squared:  0.6274
## F-statistic:  1164 on 1 and 690 DF,  p-value: < 2.2e-16
```

An increase in city-level employment by one thousand persons is associated with an increase in office rents by, on average, 2.5 cents per square metre. It is certainly debatable whether this seems small or not.

But what can we say instead about the economic interpretation? We could ask, for instance: What is a typical (i.e., a one standard deviation) increase in employment? To provide an answer to this question, we can use the `sd()` function to query the standard deviation of variable *emp*. We can then use this value to calculate the response in *rent* to an increase in employment by one standard deviation, using the estimated coefficient on variable *emp* from **Model 1**.
(Note: alternatively, you can divide the value for the standard deviation by one thousand and multiply it by the coefficient estimate on variable `I(emp/1000)` from **Model 2**.)

```
sd(dat2$emp, na.rm = TRUE) # calculates standard deviation (sd)
```

```
## [1] 119468.4
```

```
pooled.ols1$coefficients[2]*119468
```

```
##      emp
## 2.964158
```

A one standard deviation increase in employment is associated with an increase in office rents by, on average, 2.96 Euro per square metre.

Another option is to log-transform the dependent and/or independent variables (see *Tutorial 3 - Workbook* to recap on applicable differences in the interpretation of coefficient estimates). Here we use a log-log specification.

Model 3: Simple pooled OLS with log transformed variables

```
pooled.ols3 <- lm(log(rent) ~ log(emp), data = dat2)
summary(pooled.ols3)
```

```
## Call:
## lm(formula = log(rent) ~ log(emp), data = dat2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.49465 -0.13072 -0.00439  0.13043  0.71272
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.846308   0.097697  -8.663   <2e-16 ***
## log(emp)     0.279106   0.008769  31.830   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.204 on 690 degrees of freedom
##   (106 observations deleted due to missingness)
## Multiple R-squared:  0.5949, Adjusted R-squared:  0.5943
## F-statistic:  1013 on 1 and 690 DF,  p-value: < 2.2e-16
```

A one percent increase in employment is associated with an increase in per-square-metre office rents by, on average, 0.28 %.

Thus far, we have ignored the time dimension in the data, effectively treating it as if it was a cross sectional data set. Anyhow, rents may have changed over time because of macroeconomic events (e.g., inflation, GDP growth), which are common to all observation units (cities). To the extent that these events/shocks are both correlated with employment and uncontrolled for, their effects will likely confound the relationship of interest, i.e., they will show up in the estimated coefficient on *emp* (omitted variable bias (OVB)).

Similarly, we have also ignored the regional dimension in the data, i.e., for each observation unit (city), there are several observations on office rents, employment etc. (at different points in time). In Models 1 through 3, the estimated coefficient on employment does not account for the possibility that cities may differ with respect to their characteristics. More precisely, if there are important (i.e., correlated with both employment and office rents) unobserved (and uncontrolled for) variables, which differ between cities (e.g., the age distribution of workforce, education attainment and skills), this may again confound the relationship of interest, leading to OVB in our estimates. In part b., we aim to (partially) address these problems of pooled OLS.

b. Use a log-log specification to model the relationship between office rents and employment. Include a full set of dummy variables (DVs) for time periods as well as observation entities in your model.

We include a full set of fixed effects for time (here: at annual level) and region (the unit of observation, here: at city level) and estimate the model by means of a least squares dummy variable (LSDV) regression, which is essentially an OLS regression with a bunch of DVs for the, so called, *two-way fixed effects*.

Model 4: LSDV regression (log-log specification of rents & employment)

```
LSDV.ols <- lm(log(rent) ~ as.factor(City) + as.factor(Jahr) + log(emp), data = dat2)
summary(LSDV.ols)
```

```
## Call:
## lm(formula = log(rent) ~ as.factor(City) + as.factor(Jahr) +
##     log(emp), data = dat2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34638 -0.05765 -0.00216  0.05428  0.33586
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -9.89432    0.97701 -10.127  < 2e-16 ***
## as.factor(City)Bochum              1.52878    0.17541   8.715  < 2e-16 ***
## as.factor(City)Bonn                1.26812    0.13201   9.606  < 2e-16 ***
## as.factor(City)Bremen              0.81868    0.12663   6.465 2.02e-10 ***
## as.factor(City)Chemnitz            1.60776    0.18113   8.876  < 2e-16 ***
## as.factor(City)Cottbus             2.24163    0.23967   9.353  < 2e-16 ***
## as.factor(City)Dortmund            1.21995    0.13769   8.860  < 2e-16 ***
## as.factor(City)Dresden             1.04362    0.13182   7.917 1.09e-14 ***
## as.factor(City)Duisburg            1.47845    0.16406   9.012  < 2e-16 ***
## as.factor(City)Düsseldorf          0.86043    0.08106  10.614  < 2e-16 ***
## as.factor(City)Frankfurt.(Main)    0.78937    0.05896  13.389  < 2e-16 ***
## as.factor(City)Freiburg.(Breisgau) 1.74358    0.18275   9.541  < 2e-16 ***
## as.factor(City)Halle.(Saale)       1.57957    0.18920   8.349 4.35e-16 ***
## as.factor(City)Hamburg             0.11019    0.03893   2.830 0.004800 **
## as.factor(City)Hannover            0.71299    0.10352   6.887 1.37e-11 ***
## as.factor(City)Karlsruhe           1.04039    0.14372   7.239 1.31e-12 ***
## as.factor(City)Kassel              1.46025    0.19151   7.625 8.98e-14 ***
## as.factor(City)Kiel                1.42609    0.17139   8.321 5.38e-16 ***
## as.factor(City)Köln                0.46118    0.07019   6.571 1.05e-10 ***
## as.factor(City)Leipzig             1.11010    0.13869   8.004 5.75e-15 ***
## as.factor(City)Leverkusen          2.13092    0.23932   8.904  < 2e-16 ***
## as.factor(City)Lübeck              1.79270    0.20982   8.544  < 2e-16 ***
## as.factor(City)Magdeburg           1.61187    0.17817   9.047  < 2e-16 ***
## as.factor(City)Mainz               1.59523    0.17250   9.248  < 2e-16 ***
## as.factor(City)München             0.36264    0.03993   9.082  < 2e-16 ***
## as.factor(City)Münster             1.24969    0.15363   8.134 2.20e-15 ***
## as.factor(City)Neuss.(Stadt)       2.37859    0.23578  10.088  < 2e-16 ***
## as.factor(City)Nürnberg            0.64458    0.11283   5.713 1.71e-08 ***
## as.factor(City)Oberhausen          2.15907    0.23353   9.245  < 2e-16 ***
## as.factor(City)Potsdam             2.10852    0.20857  10.109  < 2e-16 ***
## as.factor(City)Regensburg          1.76341    0.19891   8.865  < 2e-16 ***
## as.factor(City)Rostock             1.99015    0.20558   9.681  < 2e-16 ***
## as.factor(City)Saarbrücken.(Stadt) 1.46569    0.18537   7.907 1.18e-14 ***
## as.factor(City)Schwerin            2.14307    0.23190   9.241  < 2e-16 ***
## as.factor(City)Stuttgart           0.68243    0.08055   8.472  < 2e-16 ***
## as.factor(City)Ulm                 2.05074    0.21356   9.603  < 2e-16 ***
## as.factor(City)Wiesbaden           1.49769    0.15336   9.766  < 2e-16 ***
## as.factor(City)Wilhelmshaven       2.46043    0.27000   9.113  < 2e-16 ***
## as.factor(City)Wolfsburg           2.32454    0.23225  10.009  < 2e-16 ***
```

```
## as.factor(City)Wuppertal            1.38299    0.17230   8.027 4.88e-15 ***
## as.factor(City)Würzburg             1.88655    0.20754   9.090  < 2e-16 ***
## as.factor(City)Zwickau              2.38247    0.25214   9.449  < 2e-16 ***
## as.factor(Jahr)1992                 0.05484    0.02353   2.331 0.020063 *
## as.factor(Jahr)1993                 0.13152    0.02227   5.906 5.73e-09 ***
## as.factor(Jahr)1994                 0.07657    0.02234   3.427 0.000650 ***
## as.factor(Jahr)1995                 0.03662    0.02212   1.655 0.098375 .
## as.factor(Jahr)1996                -0.01181    0.02208  -0.535 0.593020
## as.factor(Jahr)1997                -0.06417    0.02204  -2.911 0.003728 **
## as.factor(Jahr)1998                -0.08500    0.02204  -3.857 0.000126 ***
## as.factor(Jahr)1999                -0.09916    0.02208  -4.491 8.42e-06 ***
## as.factor(Jahr)2000                -0.09819    0.02222  -4.419 1.17e-05 ***
## as.factor(Jahr)2001                -0.08684    0.02226  -3.901 0.000106 ***
## as.factor(Jahr)2002                -0.12036    0.02226  -5.407 9.07e-08 ***
## as.factor(Jahr)2003                -0.15440    0.02219  -6.958 8.63e-12 ***
## as.factor(Jahr)2004                -0.18593    0.02219  -8.380 3.43e-16 ***
## as.factor(Jahr)2005                -0.20236    0.02219  -9.120  < 2e-16 ***
## as.factor(Jahr)2006                -0.19864    0.02224  -8.933  < 2e-16 ***
## as.factor(Jahr)2007                -0.19392    0.02244  -8.643  < 2e-16 ***
## log(emp)                            0.97108    0.07392  13.136  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09239 on 633 degrees of freedom
##   (106 observations deleted due to missingness)
## Multiple R-squared:  0.9238, Adjusted R-squared:  0.9168
## F-statistic: 132.3 on 58 and 633 DF,  p-value: < 2.2e-16
```

By including city fixed effects (FEs), we effectively control for time-invariant, (unobserved), regional heterogeneity (more generally speaking, structural differences between observation units), i.e., all variables that vary across cities, but are constant over time (the city FE).

By including year FEs, we effectively control for differences/changes in office rents over time, brought about by (unobserved) contemporaneous shocks that are common to all observation units (cities), i.e., time-varying variables that are correlated with rents and affect all cities in the same way (the year FE).

Taking region- and time-specific heterogeneity into account, the adjusted R^2 increases substantially (from 0.59 to 0.92) and the coefficient estimate on $\log(emp)$ increases by a factor of close to 3.5 (from 0.279106 to 0.97108).

c. Replicate your result from part b. using the `plm` package.

The `plm` package provides a convenient alternative to LSDV estimation and extends to further linear panel data methods, not considered in this course, e.g., first-differences estimation. The `plm` package essentially allows your computer to estimate a model using the `lm()` function as we did before, but it does so internally on transformed data according to the way we specify. To load the package and learn more about its various features, type:

```
library(plm)
vignette("A_plmPackage") #for documentation
```

First, we need to change the class of R object **dat2** from `data.frame` to `pdata.frame` ("`p`" as in

panel), such that your computer knows that we are dealing with panel data. The function to convert the data frame is simply `pdata.frame()`. In the option `index` we first specify the cross sectional identifier for the observation unit (here: variable *City*) and then the time series identifier (here: variable *Jahr* for the year FEs).

```
dat3 <- pdata.frame(dat2, index = c("City","Jahr"))
```

Once the data are set to panel data format, we can estimate our model using the `plm()` function. We specify `"within"` for the option `model`, to tell our computer to use the fixed effects (i.e., the within) estimator, which exploits variation *within* observation units (cities) over time. By specifying `"twoways"` for the option `effect`, we tell the computer to apply both region and time FEs.

Model 5: Two-way fixed effects model (log-log specification of rents & employment)

```
FE.ols <- plm(log(rent) ~ log(emp), model = "within", effect = "twoways", data = dat3)
summary(FE.ols)
```

```
## Twoways effects Within Model
## Call:
## plm(formula = log(rent) ~ log(emp), data = dat3, effect = "twoways",
##     model = "within")
##
## Unbalanced Panel: n = 42, T = 15-17, N = 692
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.3463796 -0.0576469 -0.0021647  0.0542759  0.3358607
##
## Coefficients:
##          Estimate Std. Error t-value  Pr(>|t|)
## log(emp) 0.971076   0.073925  13.136 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    6.8765
## Residual Sum of Squares: 5.4035
## R-Squared:      0.21421
## Adj. R-Squared: 0.14221
## F-statistic: 172.555 on 1 and 633 DF, p-value: < 2.22e-16
```

The output tells us that we are dealing with an unbalanced panel of 42 cross sectional units (cities), for each of which the number of pairwise observations of office rents and employment ranges between 15 to 17 annual observations. The number of observations used in the regression is 692, just as in the LSDV model.

Note that the R^2 and adjusted R^2 are much lower in this regression, despite identical coefficient estimates. This is because R^2 measures the portion of variation in the dependent variable that is explained by the model and in **Model 5**, instead of including a full set of dummies for the FEs, we consider a transformed version of the dependent variable (i.e., a demeaned version of the original rent variable), which also implies that the model's R^2 does not account for the explanatory power of the dummies that we used in the LSDV model.

d. Add further regressors to the model.

The FE model, we considered, controls for time-invariant differences between cities as well as time-varying factors that are common to all cities and correlate with office rents. However, we have not yet taken into account that there may be time-varying characteristics, which are city-specific and could potentially correlate with both office rents and employment. If we fail to control for such factors, our estimates may still suffer from OVB. Let us assume, that after careful consideration, we have concluded that the office stock (*stock*) and vacancy rate (v) are good controls, which we add to our model as follows:

Model 6: Full model example

```
# adding: i. log transformed stock (measured in sqm) (log-log model)
#         ii. vacancy rate (measured as a percentage) in levels (log-lin or semi-log model)
fm <- log(rent) ~ log(emp) + log(stock) + v  # write full model w/o FEs to "fm"

FE.ols.fm <- plm(fm,                                # run regression for model "fm"
            model = "within", effect = "twoways", # with FE settings from before
            data = dat3)                          # using dataset from before
summary(FE.ols.fm)
```

```
## Twoways effects Within Model
## Call:
## plm(formula = fm, data = dat3, effect = "twoways", model = "within")
##
## Unbalanced Panel: n = 42, T = 15-17, N = 692
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.3058304 -0.0524485 -0.0007096  0.0495026  0.2911056
##
## Coefficients:
##              Estimate Std. Error t-value  Pr(>|t|)
## log(emp)    0.8940540  0.0699386 12.7834 < 2.2e-16 ***
## log(stock) -0.6532594  0.0707335 -9.2355 < 2.2e-16 ***
## v          -0.0046510  0.0015535 -2.9939  0.002862 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    6.8765
## Residual Sum of Squares: 4.4217
## R-Squared:      0.35698
## Adj. R-Squared: 0.29584
## F-statistic: 116.771 on 3 and 631 DF, p-value: < 2.22e-16
```

When we control for these potential confounders, the coefficient on $\log(emp)$ decreases by about 0.077. All other coefficients show the expected sign and are also highly statistically significant.

A note on the interpretation of the coefficient on vacancy rate, v: this variable is included in levels as opposed to logs, because it is already measured as a percentage. An additional unit is, thus, one percentage point. The dependent variable is measured in logs (i.e., semi-log or more precisely log-level or log-linear model), which yields the following interpretation: A one %-pt increase in the vacancy rate is associated with a decrease in office rents by, on average, 0.47 %.