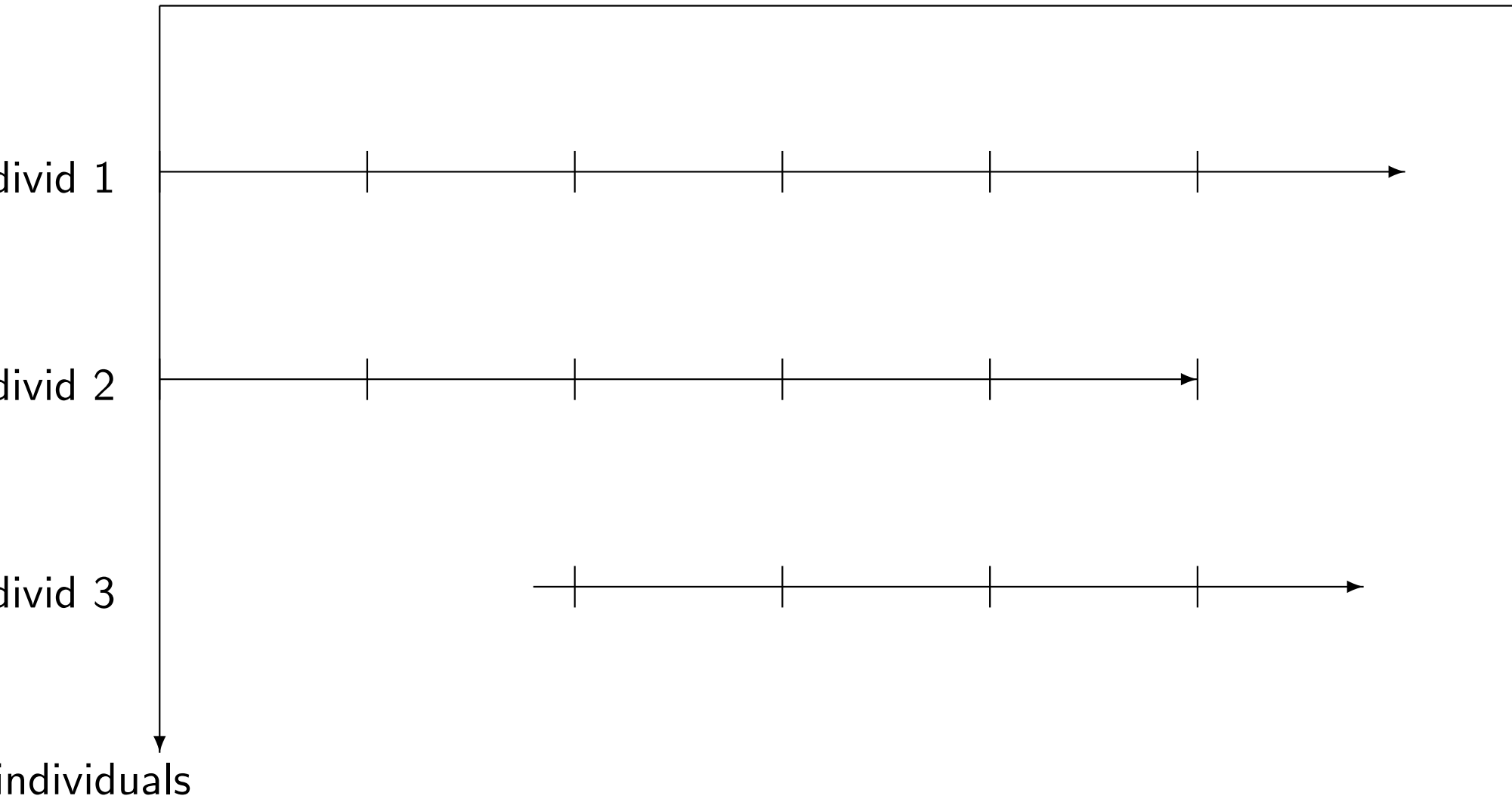
Panel data

Panel data are observations of the same individual on different dates.



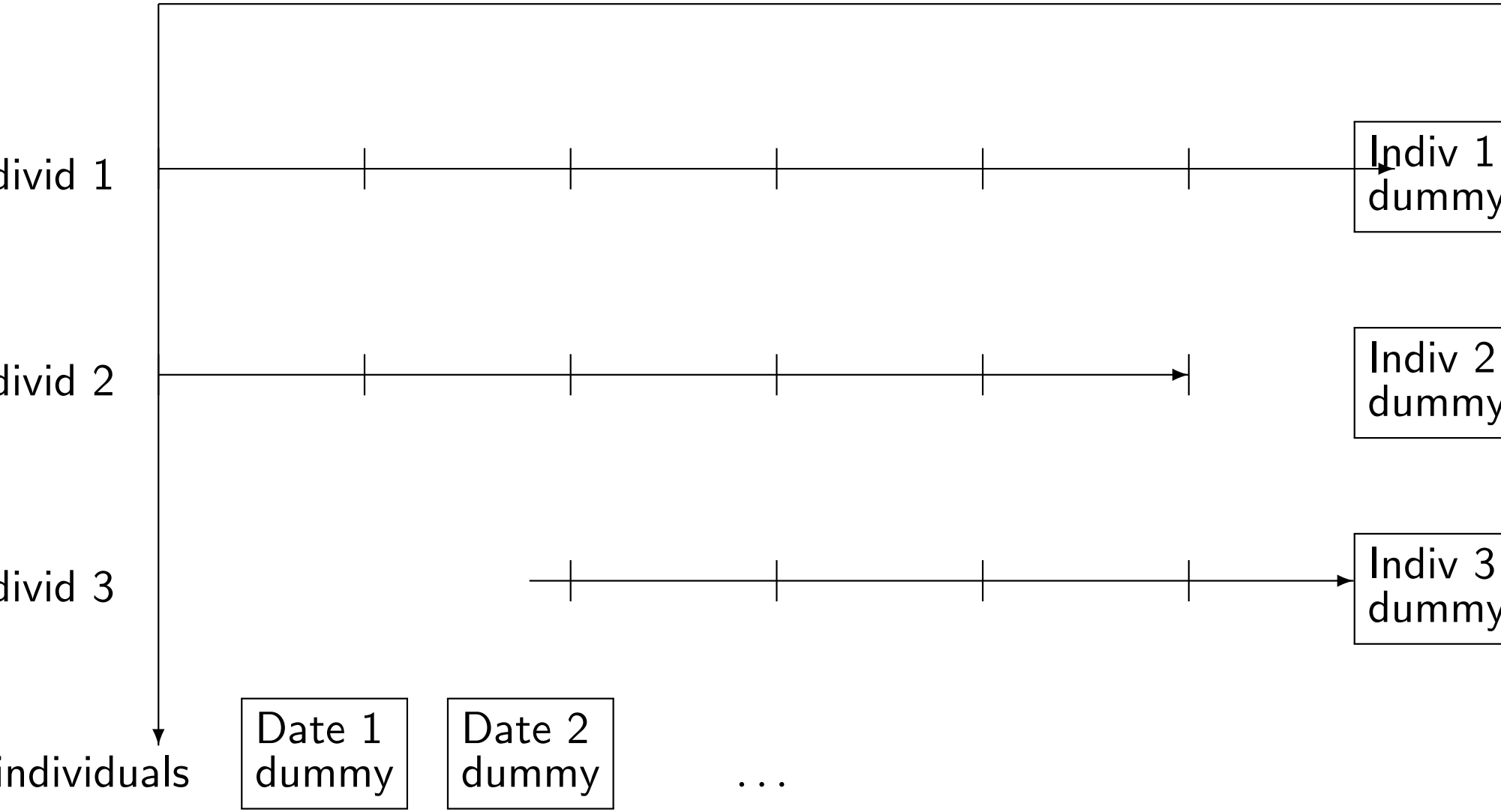
The panel is *balanced* if all individuals have a complete set of

Panel data

In such settings dummy variables can be used to control for unobserved heterogeneity
This is typically called fixed effects
We talk about

- State fixed effects
- Time fixed effects

or both


divid 1
Indiv 1
dummy
divid 2
Indiv 2
dummy
divid 3
Indiv 3
dummy
individuals
Date 1
dummy
Date 2
dummy
. . .

By including both these time and state fixed effects we control for omitted variables bias arising both from unobserved variables constant over time and from unobserved variables that are constant across states.
This is a very simple way of dealing with the panel structure of the data, but it is not the only one. Various methods have been developed that uses the panel structure in modelling.

Panel Data in R

There is a R library called `plm` which has a lot of different panel data utilitites.
We will illustrate the usage of this library.
For the students who are used to asset pricing applications, we show how we can shoehorn a standard such anaysis into the fixed effects library.

Black Jensen Scholes (1972) as a Panel

We illustrate how one can treat the estimation of the CAPM in a **?** setting. As data we use the monthly returns on five size portfolios provided by Ken French, for the period 1980–2015, together with his estimates of the risk free rate and excess market return.
Some prelimaries (Do not show reading of the data).

```
> source ("/home/bernt/data/2016/french_data/read_size_port
> source ("/home/bernt/data/2016/french_data/read_pricing_f
> library(stargazer)
> library(lmtest)
> library(plm)
```

```
> head(FFSize5EW)
             Lo.20   Qnt.2   Qnt.3   Qnt.4   Hi.20
Jul 1926 -0.0057  0.0059  0.0160  0.0147  0.0333
Aug 1926  0.0384  0.0359  0.0371  0.0161  0.0233
Sep 1926 -0.0048 -0.0140  0.0000 -0.0050 -0.0009
Oct 1926 -0.0329 -0.0410 -0.0289 -0.0336 -0.0295
Nov 1926 -0.0055  0.0218  0.0341  0.0339  0.0316
Dec 1926  0.0119  0.0378  0.0132  0.0263  0.0314
```

```
> summary(FFSize5EW)
     Index           Lo.20                Qnt.2                 C
 Min.   :1926   Min.   :-0.30990   Min.   :-0.31470   Min.
 1st Qu.:1949   1st Qu.:-0.02765   1st Qu.:-0.02445   1st Q
 Median :1971   Median : 0.01195   Median : 0.01495   Media
 Mean   :1971   Mean   : 0.01632   Mean   : 0.01256   Mean
 3rd Qu.:1994   3rd Qu.: 0.04870   3rd Qu.: 0.04775   3rd Q
 Max.   :2016   Max.   : 1.12600   Max.   : 0.81640   Max.
     Qnt.4              Hi.20
 Min.   :-0.29310   Min.   :-0.304000
 1st Qu.:-0.02038   1st Qu.:-0.019200
 Median : 0.01470   Median : 0.012300
 Mean   : 0.01079   Mean   : 0.009507
 3rd Qu.: 0.04410   3rd Qu.: 0.038875
 Max.   : 0.50640   Max.   : 0.416300
```

```
> summary(RMRF)
     Index              RMRF          
 Min.   :1926   Min.   :-0.291300  
 1st Qu.:1949   1st Qu.:-0.020275  
 Median :1971   Median : 0.010100  
 Mean   :1971   Mean   : 0.006475  
 3rd Qu.:1994   3rd Qu.: 0.036500  
 Max.   :2016   Max.   : 0.388500  
> summary(RF)
     Index               RF           
 Min.   :1926   Min.   :-0.000600  
 1st Qu.:1949   1st Qu.: 0.000400  
 Median :1971   Median : 0.002500  
 Mean   :1971   Mean   : 0.002807  
 3rd Qu.:1994   3rd Qu.: 0.004300  
 Max.   :2016   Max.   : 0.013500  
```

Pull the right subperiod, and create the excess returns

```
> data <- window(merge(FFSize5EW,RMRF,RF),
+                start=as.yearmon(1980,1),end=as.yearmon(2015,12
> Ri <- data[,1:5]
> eRi <- Ri-data$RF
> eRm <- data$RMRF
> eR1 <- eRi[,1]
> eR2 <- eRi[,2]
> eR3 <- eRi[,3]
> eR4 <- eRi[,4]
> eR5 <- eRi[,5]
```

Running OLS regressions

```
> regr1 <- lm(eR1~eRm)
> regr2 <- lm(eR2~eRm)
> regr3 <- lm(eR3~eRm)
> regr4 <- lm(eR4~eRm)
> regr5 <- lm(eR5~eRm)
> stargazer(regr1,regr2,regr3,regr4,regr5,
+           out=filename,float=FALSE,
+           omit.stat=c("f","rsq","ser"))
```

	Dependent variable:				
	eR1	eR2	eR3	eR4	eR5
	(1)	(2)	(3)	(4)	(5)
eRm	1.035^{***}	1.195^{***}	1.172^{***}	1.126^{***}	1.030^{***}
	(0.044)	(0.030)	(0.024)	(0.018)	(0.012)
Constant	0.002	-0.0002	0.0005	0.001	0.0005
	(0.002)	(0.001)	(0.001)	(0.001)	(0.001)
Observations	421	421	421	421	421
Adjusted R^2	0.573	0.789	0.850	0.908	0.945
Note:					*p<0.1; **p<0.05; ***p<0.01

Illustrate how this can be achieved withe the data organized differently.
Collect all stock returns into one long vector, together with the matching date, market return, and a portfolio indicateor (1-5).
Create a data frame with each portfolio as a separate index.

```
> portf1 <- rep(1,length(eR1))
> data1 <- data.frame(index(eR1),eR1,eRm,portf1)
> names(data1)<-c("date","eRi","eRm","portf")
> portf2 <- rep(2,length(eR2))
> data2 <- data.frame(index(eR2),eR2,eRm,portf2)
> names(data2)<-c("date","eRi","eRm","portf")

....

> PanelData <- rbind(data1,data2,data3,data4,data5)
```

```
> head(PanelData)
                date      eRi      eRm portf
Jan 1980 Jan 1980  0.1151  0.0551     1
Feb 1980 Feb 1980 -0.0081 -0.0122     1
Mar 1980 Mar 1980 -0.1871 -0.1290     1
Apr 1980 Apr 1980  0.0329  0.0397     1
May 1980 May 1980  0.0585  0.0526     1
Jun 1980 Jun 1980  0.0366  0.0306     1
```

To create a dummy for each different portfolio, the function `factor()` is useful.
Run an OLS regression

```
> portf <- PanelData$portf
> eRi <- PanelData$eRi
> eRm <- PanelData$eRm
> regrLM <- lm(eRi ~ (0 + factor(portf)) + (0+ factor(porti
```

Results

```
Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
factor(portf)1      0.0016966  0.0012583   1.348 0.177697
factor(portf)2     -0.0001937  0.0012583  -0.154 0.877663
factor(portf)3      0.0004591  0.0012583   0.365 0.715233
factor(portf)4      0.0006387  0.0012583   0.508 0.611785
factor(portf)5      0.0004982  0.0012583   0.396 0.692199
eRm                 1.0350372  0.0276948  37.373  < 2e-16 ***
factor(portf)2:eRm  0.1597144  0.0391664   4.078 4.72e-05 ***
factor(portf)3:eRm  0.1366592  0.0391664   3.489 0.000494 ***
factor(portf)4:eRm  0.0912866  0.0391664   2.331 0.019862 *
factor(portf)5:eRm -0.0047409  0.0391664  -0.121 0.903667
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02556 on 2095 degrees of freedom
Multiple R-squared:  0.7981,Adjusted R-squared:  0.7971
F-statistic: 827.9 on 10 and 2095 DF,  p-value: < 2.2e-16
```

Note:
all but the first betas are to be interpeted relative to the first beta.
But what we are after here is testing the constants, so that does not matter much.

Here is how to do the equivalent of the above using a `plm` specification.
First we do exactly the same regression as we did with `lm` above, which is done by specifying the `model="pooling"` option:

```
> regrPool <- plm(eRi ~ (0 + factor(portf)) + (0+ factor(po
+              data=PanelData,
+              model="pooling",
+              index=c("portf","date"))
```

```
Balanced Panel: n=5, T=421, N=2105

Coefficients :
                      Estimate  Std. Error t-value  Pr(>|t|)
factor(portf)1      0.00169657  0.00125827  1.3483 0.1776971
factor(portf)2     -0.00019371  0.00125827 -0.1540 0.8776632
factor(portf)3      0.00045913  0.00125827  0.3649 0.7152333
factor(portf)4      0.00063870  0.00125827  0.5076 0.6117853
factor(portf)5      0.00049819  0.00125827  0.3959 0.6921985
eRm                 1.03503719  0.02769483 37.3729 < 2.2e-16 ***
factor(portf)2:eRm  0.15971445  0.03916640  4.0778 4.715e-05 ***
factor(portf)3:eRm  0.13665919  0.03916640  3.4892 0.0004945 ***
factor(portf)4:eRm  0.09128657  0.03916640  2.3307 0.0198617 *
factor(portf)5:eRm -0.00474089  0.03916640 -0.1210 0.9036671
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-Squared:      0.79423
Adj. R-Squared: 0.79335
F-statistic: 808.623 on 10 and 2095 DF, p-value: < 2.22e-16
```

Pretty printing

```
> stargazer(regrPool,
+           float=FALSE,omit.stat=c("f","rsq","ser"))
```

	Dependent variable: eRi
factor(portf)1	0.00170
	(0.00126)
factor(portf)2	−0.00019
	(0.00126)
factor(portf)3	0.00046
	(0.00126)
factor(portf)4	0.00064
	(0.00126)
factor(portf)5	0.00050
	(0.00126)
eRm	1.03504***
	(0.02769)
factor(portf)2:eRm	0.15971***
	(0.03917)
factor(portf)3:eRm	0.13666***
	(0.03917)
factor(portf)4:eRm	0.09129**
	(0.03917)
factor(portf)5:eRm	−0.00474
	(0.03917)
Observations	2,105
Adjusted R^2	0.79335
Note:	*p<0.1; **p<0.05; ***p<0.01

Let us now let the portfolio dummies be created automatically, using the specification `model="within"`:

```
> regrFE <- plm(eRi ~ 0 + factor(portf)*eRm,
+               data=PanelData,
+               model="within",
+               index=c("portf","date"))
```

```
> summary(regrFE)
Oneway (individual) effect Within Model
Balanced Panel: n=5, T=421, N=2105

Coefficients :
                     Estimate Std. Error t-value  Pr(>|t|)
eRm                 1.0350372  0.0276948 37.3729 < 2.2e-16 ***
factor(portf)2:eRm  0.1597144  0.0391664  4.0778 4.715e-05 ***
factor(portf)3:eRm  0.1366592  0.0391664  3.4892 0.0004945 ***
factor(portf)4:eRm  0.0912866  0.0391664  2.3307 0.0198617 *
factor(portf)5:eRm -0.0047409  0.0391664 -0.1210 0.9036671
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    6.6512
Residual Sum of Squares: 1.3687
R-Squared:      0.79422
Adj. R-Squared: 0.79333
F-statistic: 1617.13 on 5 and 2095 DF, p-value: < 2.22e-16
```

The standard way of printing the summary does not include the fixed effects, but they are available separately:

```
> fe <- fixef(regrFE)
> summary(fe)
     Estimate  Std. Error t-value Pr(>|t|)
1  0.00169657  0.00125827  1.3483   0.1777
2 -0.00019371  0.00125827 -0.1540   0.8777
3  0.00045913  0.00125827  0.3649   0.7152
4  0.00063870  0.00125827  0.5076   0.6118
5  0.00049819  0.00125827  0.3959   0.6922
```

So, here you have the same conclusions about the coefficients as you had in the LM regressions

	Dependent variable:				
	eR1	eR2	eR3	eR4	eR5
	(1)	(2)	(3)	(4)	(5)
eRm	1.03504***	1.19475***	1.17170***	1.12632***	1.03030**
	(0.04354)	(0.03013)	(0.02401)	(0.01753)	(0.01217
Constant	0.00170	−0.00019	0.00046	0.00064	0.00050
	(0.00198)	(0.00137)	(0.00109)	(0.00080)	(0.00055
Observations	421	421	421	421	421
Adjusted R^2	0.57321	0.78914	0.85005	0.90764	0.94468
Note:					*p<0.1; **p<0.05; ***p<0.0

The only difference in this case is that we have imposed that the std error is the same across stocks.

Well, that was not the best example of a fixed effects regression, as it is not really one, it is more showing how the fixed effects commands work for people familiar with asset pricing investigations.