Exercise 3 (2025) — Advanced Methods for Regression and Classification

Olesia Galynskaia 12321492

2025-10-29

Loading and observing data

```
load("building.RData")
stopifnot(exists("df"), is.data.frame(df))
attributes(df)
```

```
## $names
##   [1] "y"                  "START.YEAR"         "START.QUARTER"
##   [4] "COMPLETION.YEAR"    "COMPLETION.QUARTER" "PhysFin1"
##   [7] "PhysFin2"           "PhysFin3"           "PhysFin4"
##  [10] "PhysFin5"           "PhysFin6"           "PhysFin7"
##  [13] "PhysFin8"           "Econ1"              "Econ2"
##  [16] "Econ3"              "Econ4"              "Econ5"
##  [19] "Econ6"              "Econ7"              "Econ8"
##  [22] "Econ9"              "Econ10"             "Econ11"
##  [25] "Econ12"             "Econ13"             "Econ14"
##  [28] "Econ15"             "Econ16"             "Econ17"
##  [31] "Econ18"             "Econ19"             "Econ1.lag1"
##  [34] "Econ2.lag1"         "Econ3.lag1"         "Econ4.lag1"
##  [37] "Econ5.lag1"         "Econ6.lag1"         "Econ7.lag1"
##  [40] "Econ8.lag1"         "Econ9.lag1"         "Econ10.lag1"
##  [43] "Econ11.lag1"        "Econ12.lag1"        "Econ13.lag1"
##  [46] "Econ14.lag1"        "Econ15.lag1"        "Econ16.lag1"
##  [49] "Econ17.lag1"        "Econ18.lag1"        "Econ19.lag1"
##  [52] "Econ1.lag2"         "Econ2.lag2"         "Econ3.lag2"
##  [55] "Econ4.lag2"         "Econ5.lag2"         "Econ6.lag2"
##  [58] "Econ7.lag2"         "Econ8.lag2"         "Econ9.lag2"
##  [61] "Econ10.lag2"        "Econ11.lag2"        "Econ12.lag2"
##  [64] "Econ13.lag2"        "Econ14.lag2"        "Econ15.lag2"
##  [67] "Econ16.lag2"        "Econ17.lag2"        "Econ18.lag2"
##  [70] "Econ19.lag2"        "Econ1.lag3"         "Econ2.lag3"
##  [73] "Econ3.lag3"         "Econ4.lag3"         "Econ5.lag3"
##  [76] "Econ6.lag3"         "Econ7.lag3"         "Econ8.lag3"
```

```
## [79] "Econ9.lag3"          "Econ10.lag3"         "Econ11.lag3"
## [82] "Econ12.lag3"         "Econ13.lag3"         "Econ14.lag3"
## [85] "Econ15.lag3"         "Econ16.lag3"         "Econ17.lag3"
## [88] "Econ18.lag3"         "Econ19.lag3"         "Econ1.lag4"
## [91] "Econ2.lag4"          "Econ3.lag4"          "Econ4.lag4"
## [94] "Econ5.lag4"          "Econ6.lag4"          "Econ7.lag4"
## [97] "Econ8.lag4"          "Econ9.lag4"          "Econ10.lag4"
## [100] "Econ11.lag4"         "Econ12.lag4"         "Econ13.lag4"
## [103] "Econ14.lag4"         "Econ15.lag4"         "Econ16.lag4"
## [106] "Econ17.lag4"         "Econ18.lag4"         "Econ19.lag4"
##
## $class
## [1] "data.frame"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
## [145] 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
## [163] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
## [181] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
## [199] 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
## [217] 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
## [235] 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
## [253] 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
## [271] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
## [289] 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
## [307] 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
## [325] 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
## [343] 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
## [361] 361 362 363 364 365 366 367 368 369 370 371 372
```

```r
str(attributes(df))
```

```
## List of 3
##  $ names    : chr [1:108] "y" "START.YEAR" "START.QUARTER" "COMPLETION.YEAR" ...
##  $ class    : chr "data.frame"
##  $ row.names: int [1:372] 1 2 3 4 5 6 7 8 9 10 ...
```

```r
head(sapply(df, function(x) attr(x, "label")), 10)
```

```
## $y
```

```
## NULL
##
## $START.YEAR
## NULL
##
## $START.QUARTER
## NULL
##
## $COMPLETION.YEAR
## NULL
##
## $COMPLETION.QUARTER
## NULL
##
## $PhysFin1
## NULL
##
## $PhysFin2
## NULL
##
## $PhysFin3
## NULL
##
## $PhysFin4
## NULL
##
## $PhysFin5
## NULL
```

Quick structure and summary

```r
cat("**Dimensions:**", nrow(df), "rows ×", ncol(df), "columns\n\n")
```

```
## **Dimensions:** 372 rows × 108 columns
```

```r
str(df[, 1:10])
```

```
## 'data.frame':    372 obs. of  10 variables:
##  $ y                 : num  7.7 8.52 7.09 5.11 8.61 ...
##  $ START.YEAR        : num  81 84 78 72 87 87 87 88 76 80 ...
##  $ START.QUARTER     : num  1 1 1 2 1 1 2 1 3 1 ...
##  $ COMPLETION.YEAR   : num  85 89 81 73 90 90 90 89 77 80 ...
##  $ COMPLETION.QUARTER: num  1 4 4 2 2 1 1 3 4 4 ...
##  $ PhysFin1          : num  1 1 1 1 1 1 1 1 1 1 ...
```

```
## $ PhysFin2          : num  3150 7600 4800 685 3000 2500 1810 1150 2110 3030 ...
## $ PhysFin3          : num  920 1140 840 202 800 640 492 380 540 930 ...
## $ PhysFin4          : num  598.5 3040 480 13.7 1230 ...
## $ PhysFin5          : num  190 400 100 20 410 420 640 500 90 170 ...
```

```r
summary(df)
```

```
##        y           START.YEAR    START.QUARTER   COMPLETION.YEAR
## Min.   :3.912   Min.   :72.00   Min.   :1.000   Min.   :73.00
## 1st Qu.:6.359   1st Qu.:78.00   1st Qu.:1.000   1st Qu.:80.00
## Median :6.908   Median :82.00   Median :2.000   Median :84.00
## Mean   :6.902   Mean   :81.48   Mean   :2.191   Mean   :82.95
## 3rd Qu.:7.438   3rd Qu.:85.00   3rd Qu.:3.000   3rd Qu.:87.00
## Max.   :8.825   Max.   :88.00   Max.   :4.000   Max.   :90.00
## COMPLETION.QUARTER    PhysFin1         PhysFin2        PhysFin3
## Min.   :1.000      Min.   : 1.000   Min.   :  200   Min.   :  60.0
## 1st Qu.:2.000      1st Qu.: 4.000   1st Qu.:  720   1st Qu.: 190.0
## Median :3.000      Median : 8.000   Median : 1220   Median : 300.0
## Mean   :2.586      Mean   : 9.728   Mean   : 1729   Mean   : 426.1
## 3rd Qu.:4.000      3rd Qu.:17.000   3rd Qu.: 2100   3rd Qu.: 490.5
## Max.   :4.000      Max.   :20.000   Max.   :15670   Max.   :5000.0
##    PhysFin4          PhysFin5        PhysFin6         PhysFin7
## Min.   :   3.7   Min.   : 10.0   Min.   : 193.1   Min.   : 2.000
## 1st Qu.:  67.8   1st Qu.: 80.0   1st Qu.: 391.7   1st Qu.: 5.000
## Median : 164.7   Median :140.0   Median : 522.5   Median : 6.000
## Mean   : 327.9   Mean   :163.1   Mean   : 554.4   Mean   : 6.266
## 3rd Qu.: 366.1   3rd Qu.:230.0   3rd Qu.: 667.9   3rd Qu.: 7.000
## Max.   :7208.2   Max.   :640.0   Max.   :3436.9   Max.   :23.000
##    PhysFin8        Econ1          Econ2            Econ3
## Min.   :  40   Min.   :1562   Min.   : 12.10   Min.   : 10.03
## 1st Qu.: 440   1st Qu.:2842   1st Qu.: 45.60   1st Qu.: 51.63
## Median : 805   Median :3629   Median : 74.90   Median : 79.28
## Mean   :1088   Mean   :4211   Mean   : 94.43   Mean   : 88.05
## 3rd Qu.:1300   3rd Qu.:6024   3rd Qu.:137.40   3rd Qu.:125.83
## Max.   :5700   Max.   :7196   Max.   :274.00   Max.   :225.00
##     Econ4           Econ5             Econ6             Econ7
## Min.   :0.920   Min.   :  38194   Min.   :  287.2   Min.   : 13.60
## 1st Qu.:2.470   1st Qu.: 183726   1st Qu.: 1979.0   1st Qu.: 39.70
## Median :3.250   Median : 445458   Median : 3819.0   Median : 87.05
## Mean   :3.605   Mean   : 641112   Mean   : 4805.6   Mean   : 98.68
## 3rd Qu.:4.720   3rd Qu.:1059966   3rd Qu.: 6622.5   3rd Qu.:117.40
## Max.   :6.880   Max.   :2171923   Max.   :18690.9   Max.   :319.38
##     Econ8            Econ9             Econ10          Econ11
## Min.   : 17.03   Min.   :  154.4   Min.   :11.00   Min.   : 170.3
## 1st Qu.: 93.00   1st Qu.: 3622.2   1st Qu.:14.00   1st Qu.: 641.5
## Median :162.75   Median :10445.6   Median :15.00   Median :1023.7
## Mean   :182.00   Mean   :18861.3   Mean   :14.07   Mean   :1327.5
```

```
##  3rd Qu.:242.27   3rd Qu.:21723.4   3rd Qu.:15.00   3rd Qu.:1994.6
##  Max.   :432.40   Max.   :73143.5   Max.   :15.00   Max.   :4188.6
##      Econ12           Econ13         Econ14          Econ15
##  Min.   : 211.1   Min.   :1592   Min.   : 1601   Min.   : 11.62
##  1st Qu.: 744.5   1st Qu.:1755   1st Qu.: 8001   1st Qu.: 51.89
##  Median :1203.3   Median :8210   Median : 8393   Median : 84.46
##  Mean   :1466.3   Mean   :5934   Mean   : 7805   Mean   : 88.38
##  3rd Qu.:2025.0   3rd Qu.:9138   3rd Qu.: 9208   3rd Qu.:123.37
##  Max.   :4741.6   Max.   :9967   Max.   :10099   Max.   :204.70
##      Econ16           Econ17           Econ18          Econ19
##  Min.   : 10.06   Min.   :  354.6   Min.   : 8436   Min.   : 141543
##  1st Qu.: 42.87   1st Qu.: 2134.5   1st Qu.:12393   1st Qu.: 588021
##  Median : 81.47   Median : 7334.8   Median :26438   Median : 825511
##  Mean   : 87.07   Mean   : 6604.9   Mean   :28297   Mean   :1041556
##  3rd Qu.:127.33   3rd Qu.:10082.0   3rd Qu.:41407   3rd Qu.:1660444
##  Max.   :222.60   Max.   :13596.4   Max.   :50928   Max.   :2606321
##    Econ1.lag1     Econ2.lag1       Econ3.lag1       Econ4.lag1
##  Min.   :1562   Min.   : 11.60   Min.   :  8.50   Min.   :0.920
##  1st Qu.:2734   1st Qu.: 44.50   1st Qu.: 49.80   1st Qu.:2.440
##  Median :3561   Median : 71.15   Median : 77.46   Median :3.150
##  Mean   :3990   Mean   : 89.63   Mean   : 84.38   Mean   :3.413
##  3rd Qu.:5606   3rd Qu.:130.50   3rd Qu.:117.05   3rd Qu.:4.300
##  Max.   :7196   Max.   :267.80   Max.   :225.00   Max.   :6.880
##    Econ5.lag1        Econ6.lag1        Econ7.lag1       Econ8.lag1
##  Min.   :  35859   Min.   :  287.2   Min.   : 12.67   Min.   : 17.03
##  1st Qu.: 176543   1st Qu.: 1861.2   1st Qu.: 35.00   1st Qu.: 98.33
##  Median : 422306   Median : 3663.5   Median : 83.80   Median :167.05
##  Mean   : 600257   Mean   : 4594.8   Mean   : 92.15   Mean   :186.69
##  3rd Qu.: 961139   3rd Qu.: 5146.3   3rd Qu.:112.80   3rd Qu.:252.88
##  Max.   :2116614   Max.   :18690.9   Max.   :306.93   Max.   :432.40
##    Econ9.lag1       Econ10.lag1     Econ11.lag1      Econ12.lag1
##  Min.   :  154.4   Min.   :11.00   Min.   : 165.1   Min.   : 208.6
##  1st Qu.: 3622.2   1st Qu.:14.00   1st Qu.: 627.6   1st Qu.: 717.9
##  Median :10866.5   Median :15.00   Median :1010.0   Median :1176.5
##  Mean   :18415.3   Mean   :14.18   Mean   :1249.0   Mean   :1385.7
##  3rd Qu.:21723.4   3rd Qu.:15.00   3rd Qu.:1821.6   3rd Qu.:1932.5
##  Max.   :73143.5   Max.   :15.00   Max.   :3962.2   Max.   :4472.3
##   Econ13.lag1    Econ14.lag1     Econ15.lag1      Econ16.lag1
##  Min.   :1504   Min.   : 1582   Min.   : 10.86   Min.   :  9.79
##  1st Qu.:1755   1st Qu.: 7994   1st Qu.: 50.28   1st Qu.: 41.80
##  Median :8075   Median : 8382   Median : 81.60   Median : 78.48
##  Mean   :5724   Mean   : 7714   Mean   : 84.91   Mean   : 83.43
##  3rd Qu.:9133   3rd Qu.: 9168   3rd Qu.:120.24   3rd Qu.:121.94
##  Max.   :9967   Max.   :10099   Max.   :201.66   Max.   :218.40
##   Econ17.lag1       Econ18.lag1     Econ19.lag1       Econ1.lag2
##  Min.   :  354.6   Min.   : 8436   Min.   : 129102   Min.   :1562
##  1st Qu.: 2000.4   1st Qu.:18967   1st Qu.: 566492   1st Qu.:2700
##  Median : 5900.0   Median :31940   Median : 802773   Median :3561
```

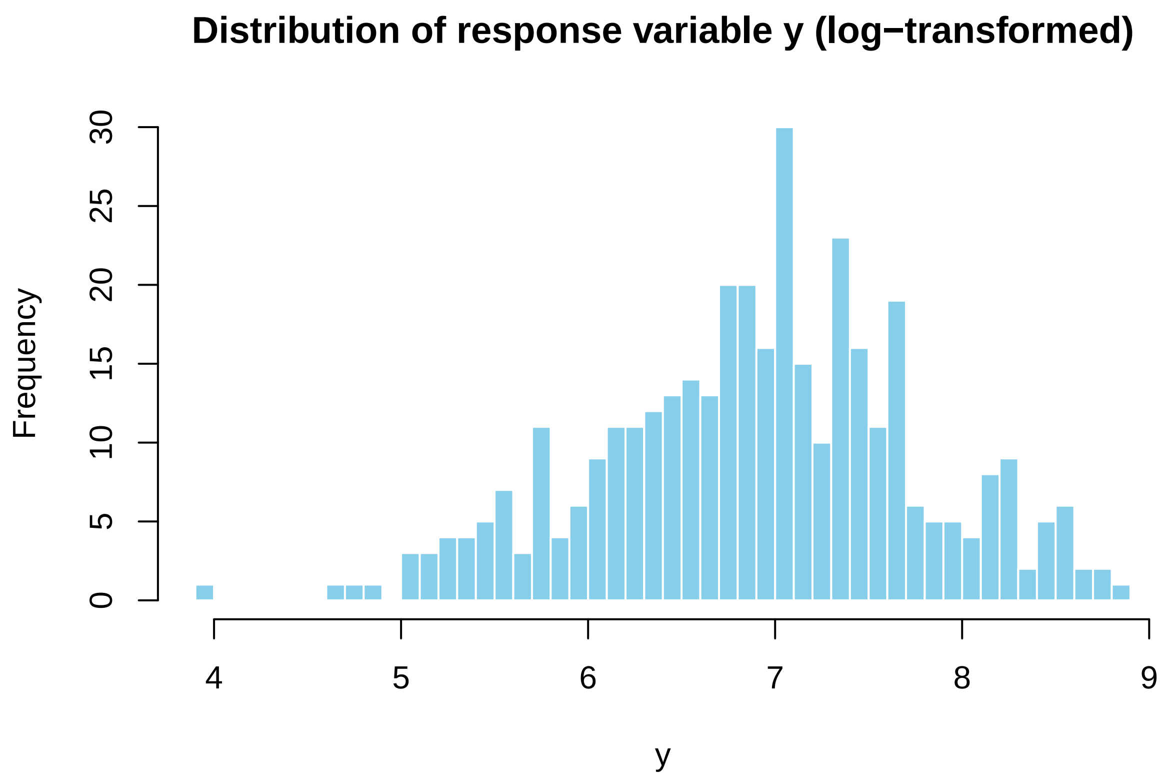
```
##  Mean   : 6462.1   Mean   :29170   Mean   : 987881   Mean   :3886
##  3rd Qu.:10082.0   3rd Qu.:37179   3rd Qu.:1654038   3rd Qu.:4986
##  Max.   :13596.4   Max.   :50928   Max.   :2435004   Max.   :7196
##    Econ2.lag2        Econ3.lag2        Econ4.lag2     Econ5.lag2
##  Min.   : 11.40   Min.   :  6.97   Min.   :0.92   Min.   :  32794
##  1st Qu.: 43.40   1st Qu.: 46.94   1st Qu.:2.45   1st Qu.: 166267
##  Median : 67.80   Median : 74.71   Median :3.05   Median : 399813
##  Mean   : 85.21   Mean   : 81.18   Mean   :3.36   Mean   : 563182
##  3rd Qu.:124.40   3rd Qu.:108.42   3rd Qu.:3.94   3rd Qu.: 921019
##  Max.   :261.50   Max.   :225.00   Max.   :6.88   Max.   :1970485
##    Econ6.lag2         Econ7.lag2        Econ8.lag2        Econ9.lag2
##  Min.   :  287.2   Min.   : 11.73   Min.   : 17.03   Min.   :  154.4
##  1st Qu.: 1668.9   1st Qu.: 34.70   1st Qu.:104.40   1st Qu.: 3994.7
##  Median : 3755.8   Median : 79.30   Median :167.05   Median : 9342.5
##  Mean   : 4238.8   Mean   : 86.49   Mean   :174.29   Mean   :16370.4
##  3rd Qu.: 5138.6   3rd Qu.:110.30   3rd Qu.:217.00   3rd Qu.:21723.4
##  Max.   :18690.9   Max.   :306.70   Max.   :432.40   Max.   :73143.5
##   Econ10.lag2     Econ11.lag2      Econ12.lag2      Econ13.lag2
##  Min.   :11.00   Min.   : 165.1   Min.   : 208.6   Min.   :1450
##  1st Qu.:14.00   1st Qu.: 627.6   1st Qu.: 680.3   1st Qu.:1755
##  Median :15.00   Median : 956.0   Median :1054.7   Median :7990
##  Mean   :14.26   Mean   :1189.0   Mean   :1314.7   Mean   :5590
##  3rd Qu.:15.00   3rd Qu.:1821.6   3rd Qu.:1932.5   3rd Qu.:9114
##  Max.   :15.00   Max.   :3746.0   Max.   :4215.9   Max.   :9967
##   Econ14.lag2     Econ15.lag2      Econ16.lag2      Econ17.lag2
##  Min.   : 1507   Min.   : 10.17   Min.   :  9.35   Min.   :  354.6
##  1st Qu.: 7994   1st Qu.: 49.92   1st Qu.: 40.26   1st Qu.: 1976.3
##  Median : 8354   Median : 77.53   Median : 75.29   Median : 5097.0
##  Mean   : 7623   Mean   : 81.66   Mean   : 79.71   Mean   : 6349.5
##  3rd Qu.: 9131   3rd Qu.:116.56   3rd Qu.:119.13   3rd Qu.:10149.0
##  Max.   :10099   Max.   :196.76   Max.   :215.00   Max.   :13596.4
##   Econ18.lag2     Econ19.lag2        Econ1.lag3     Econ2.lag3
##  Min.   : 8436   Min.   : 123618   Min.   :1562   Min.   : 10.60
##  1st Qu.:20704   1st Qu.: 540681   1st Qu.:2647   1st Qu.: 41.00
##  Median :24786   Median : 740309   Median :3321   Median : 64.40
##  Mean   :27456   Mean   : 939045   Mean   :3866   Mean   : 80.76
##  3rd Qu.:36195   3rd Qu.:1391757   3rd Qu.:4986   3rd Qu.:120.20
##  Max.   :50928   Max.   :2435004   Max.   :7196   Max.   :259.50
##    Econ3.lag3        Econ4.lag3       Econ5.lag3        Econ6.lag3
##  Min.   :  5.44   Min.   :0.920   Min.   :  30013   Min.   :  287.2
##  1st Qu.: 41.25   1st Qu.:2.320   1st Qu.: 160402   1st Qu.: 1571.1
##  Median : 71.94   Median :2.945   Median : 373644   Median : 3755.8
##  Mean   : 78.06   Mean   :3.193   Mean   : 525388   Mean   : 3944.4
##  3rd Qu.:107.20   3rd Qu.:3.720   3rd Qu.: 832124   3rd Qu.: 5131.4
##  Max.   :225.00   Max.   :6.880   Max.   :1901366   Max.   :18690.9
##    Econ7.lag3        Econ8.lag3        Econ9.lag3        Econ10.lag3
##  Min.   : 10.79   Min.   : 17.03   Min.   :  154.4   Min.   :11.00
##  1st Qu.: 34.40   1st Qu.: 74.85   1st Qu.: 2996.0   1st Qu.:14.00
```

```
##  Median : 75.60   Median :119.75   Median : 7834.2   Median :15.00
##  Mean   : 81.54   Mean   :145.84   Mean   :13351.0   Mean   :14.31
##  3rd Qu.:109.60   3rd Qu.:208.80   3rd Qu.:17361.2   3rd Qu.:15.00
##  Max.   :306.70   Max.   :432.40   Max.   :73143.5   Max.   :15.00
##   Econ11.lag3      Econ12.lag3      Econ13.lag3    Econ14.lag3
##  Min.   : 165.1   Min.   : 158.4   Min.   :1439   Min.   : 1450
##  1st Qu.: 611.8   1st Qu.: 677.7   1st Qu.:1755   1st Qu.: 7773
##  Median : 896.8   Median : 971.5   Median :7954   Median : 8325
##  Mean   :1140.1   Mean   :1245.4   Mean   :5522   Mean   : 7537
##  3rd Qu.:1763.9   3rd Qu.:1837.4   3rd Qu.:9064   3rd Qu.: 9078
##  Max.   :3499.4   Max.   :3823.6   Max.   :9967   Max.   :10099
##   Econ15.lag3      Econ16.lag3      Econ17.lag3       Econ18.lag3
##  Min.   :  9.91   Min.   :  8.85   Min.   :  354.6   Min.   : 8436
##  1st Qu.: 45.91   1st Qu.: 38.34   1st Qu.: 1966.4   1st Qu.:11774
##  Median : 74.50   Median : 71.46   Median : 4909.7   Median :21855
##  Mean   : 78.93   Mean   : 76.32   Mean   : 6131.1   Mean   :23470
##  3rd Qu.:112.15   3rd Qu.:115.70   3rd Qu.:10078.4   3rd Qu.:32783
##  Max.   :191.63   Max.   :212.10   Max.   :13596.4   Max.   :50928
##   Econ19.lag3        Econ1.lag4     Econ2.lag4      Econ3.lag4
##  Min.   : 121857   Min.   :1381   Min.   : 10.00   Min.   :  3.91
##  1st Qu.: 524765   1st Qu.:2641   1st Qu.: 40.30   1st Qu.: 40.84
##  Median : 681120   Median :3255   Median : 60.85   Median : 68.18
##  Mean   : 910297   Mean   :3757   Mean   : 76.65   Mean   : 74.52
##  3rd Qu.:1183641   3rd Qu.:4691   3rd Qu.:116.30   3rd Qu.:104.71
##  Max.   :2435004   Max.   :7196   Max.   :255.80   Max.   :225.00
##    Econ4.lag4    Econ5.lag4        Econ6.lag4       Econ7.lag4
##  Min.   :0.92   Min.   :  27231   Min.   :  287.2   Min.   :  9.85
##  1st Qu.:2.44   1st Qu.: 150267   1st Qu.: 1554.8   1st Qu.: 34.10
##  Median :2.84   Median : 352256   Median : 3485.8   Median : 72.25
##  Mean   :3.16   Mean   : 493874   Mean   : 3588.1   Mean   : 76.56
##  3rd Qu.:3.56   3rd Qu.: 784949   3rd Qu.: 4730.8   3rd Qu.:109.10
##  Max.   :6.88   Max.   :1704944   Max.   :18690.9   Max.   :306.70
##    Econ8.lag4       Econ9.lag4       Econ10.lag4     Econ11.lag4
##  Min.   : 14.15   Min.   :  152.6   Min.   :11.00   Min.   : 165.1
##  1st Qu.: 83.70   1st Qu.: 2967.7   1st Qu.:14.00   1st Qu.: 614.0
##  Median :148.80   Median : 7874.4   Median :15.00   Median : 859.1
##  Mean   :174.59   Mean   :15297.0   Mean   :14.45   Mean   :1082.0
##  3rd Qu.:251.10   3rd Qu.:17584.3   3rd Qu.:15.00   3rd Qu.:1534.6
##  Max.   :432.40   Max.   :73143.5   Max.   :15.00   Max.   :3447.8
##   Econ12.lag4      Econ13.lag4    Econ14.lag4     Econ15.lag4
##  Min.   : 152.2   Min.   :1439   Min.   : 1450   Min.   :  9.73
##  1st Qu.: 669.8   1st Qu.:1755   1st Qu.: 6714   1st Qu.: 43.40
##  Median : 938.4   Median :7928   Median : 8315   Median : 72.56
##  Mean   :1187.5   Mean   :5403   Mean   : 7432   Mean   : 76.29
##  3rd Qu.:1795.3   3rd Qu.:9001   3rd Qu.: 9022   3rd Qu.:109.02
##  Max.   :3686.3   Max.   :9967   Max.   :10099   Max.   :190.50
##   Econ16.lag4      Econ17.lag4       Econ18.lag4     Econ19.lag4
##  Min.   :  8.34   Min.   :  354.6   Min.   : 8194   Min.   : 121857
```

```
##   1st Qu.: 36.45   1st Qu.: 1917.4   1st Qu.:12065   1st Qu.: 519680
##   Median : 67.45   Median : 4525.4   Median :25759   Median : 659243
##   Mean   : 73.45   Mean   : 5915.6   Mean   :27552   Mean   : 878971
##   3rd Qu.:112.00   3rd Qu.: 9821.0   3rd Qu.:40234   3rd Qu.:1181856
##   Max.   :204.80   Max.   :13596.4   Max.   :49572   Max.   :2435004
```

Plot of response variable

```r
hist(df$y, breaks = 40,
     main = "Distribution of response variable y (log-transformed)",
     xlab = "y", col = "skyblue", border = "white")
```




```r
# compute correlations
num_data <- df[, sapply(df, is.numeric)]
cor_mat <- cor(num_data, use = "pairwise.complete.obs")

# keep only pairs with |r| > 0.9
high_corr <- which(abs(cor_mat) > 0.9 & abs(cor_mat) < 1, arr.ind = TRUE)
corr_pairs <- unique(t(apply(high_corr, 1, sort)))

cat("Highly correlated pairs (|r| > 0.9):\n")
```

```
## Highly correlated pairs (|r| > 0.9):
```

```r
print(head(data.frame(
  Var1 = rownames(cor_mat)[corr_pairs[,1]],
  Var2 = colnames(cor_mat)[corr_pairs[,2]],
  r = round(cor_mat[corr_pairs], 3)
), 10))
```

```
##          Var1            Var2     r
## 1  START.YEAR COMPLETION.YEAR 0.988
## 2  START.YEAR           Econ2 0.905
## 3  START.YEAR           Econ3 0.934
## 4  START.YEAR          Econ11 0.909
## 5  START.YEAR          Econ14 0.900
## 6  START.YEAR          Econ15 0.965
## 7  START.YEAR          Econ16 0.956
## 8  START.YEAR          Econ19 0.902
## 9  START.YEAR      Econ2.lag1 0.908
## 10 START.YEAR      Econ3.lag1 0.939
```

Data preparation

```r
# 1) Train/Test split (2/3 : 1/3) - clean and reproducible
set.seed(12321492)  # for reproducibility
stopifnot(exists("df"), is.data.frame(df), "y" %in% names(df))

n <- nrow(df)
idx_train <- sample(seq_len(n), size = floor(2/3 * n))

train <- df[idx_train, , drop = FALSE]
test  <- df[-idx_train, , drop = FALSE]

# function for RMSE
rmse <- function(actual, predicted) sqrt(mean((actual - predicted)^2))

# short info output
cat("Train:", nrow(train), "rows | Test:", nrow(test), "rows\n")
```

```
## Train: 248 rows | Test: 124 rows
```

Ex-1

(1a) Fit PCR on the training set with 10-fold CV and scaling

```r
# Packages
library(pls)  # pcr(), RMSEP(), predplot(), validationplot()

# Predictors = all columns except y
predictors <- setdiff(names(train), "y")

# Fit PCR with 10-fold cross-validation and scaling
set.seed(12321492)
fit_pcr <- pcr(
  y ~ .,
  data       = train[, c("y", predictors)],
  scale      = TRUE,
  validation = "CV",
  segments   = 10
)

# Model summary (number of comps, variance explained, etc.)
summary(fit_pcr)
```

```
## Data:    X dimension: 248 107
##  Y dimension: 248 1
## Fit method: svdpc
## Number of components considered: 107
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8573   0.5785   0.5790   0.5517   0.5207   0.4955   0.4239
## adjCV       0.8573   0.5783   0.5788   0.5478   0.5201   0.4977   0.4224
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.4231   0.4141   0.3636    0.3642    0.3662    0.3664    0.3621
## adjCV   0.4214   0.4120   0.3611    0.3619    0.3643    0.3646    0.3603
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.3559    0.3489    0.3342    0.3184    0.3026    0.3112    0.3087
## adjCV    0.3542    0.3462    0.3263    0.3145    0.3002    0.3086    0.3058
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV       0.3103    0.3104    0.3150    0.3157    0.3164    0.3213    0.3237
## adjCV    0.3077    0.3074    0.3119    0.3129    0.3131    0.3179    0.3202
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV       0.3164    0.3198     0.310    0.2972    0.2991    0.2997    0.2974
## adjCV    0.3124    0.3165     0.306    0.2928    0.2947    0.2956    0.2942
##        35 comps  36 comps  37 comps  38 comps  39 comps  40 comps  41 comps
```

```
## CV       0.2985    0.2915    0.2903    0.2903    0.2944    0.2896    0.2903
## adjCV    0.2953    0.2860    0.2863    0.2865    0.2909    0.2859    0.2854
##        42 comps  43 comps  44 comps  45 comps  46 comps  47 comps  48 comps
## CV       0.2923    0.2896    0.2946    0.2939    0.2952    0.3032    0.3044
## adjCV    0.2880    0.2853    0.2905    0.2889    0.2905    0.2981    0.2995
##        49 comps  50 comps  51 comps  52 comps  53 comps  54 comps  55 comps
## CV       0.3014    0.3039    0.3039    0.3059    0.3071    0.3101    0.3107
## adjCV    0.2976    0.2987    0.2983    0.3003    0.3015    0.3044    0.3048
##        56 comps  57 comps  58 comps  59 comps  60 comps  61 comps  62 comps
## CV       0.3122    0.3147    0.3141    0.3038    0.3030    0.3007    0.3010
## adjCV    0.3063    0.3088    0.3080    0.2987    0.2973    0.2953    0.2953
##        63 comps  64 comps  65 comps  66 comps  67 comps  68 comps  69 comps
## CV       0.3052    0.3084    0.3189    0.3328    0.3361    0.3298    0.3338
## adjCV    0.2996    0.3024    0.3123    0.3260    0.3301    0.3209    0.3249
##         70 comps   71 comps   72 comps   73 comps   74 comps  75 comps
## CV     5.629e+10  1.687e+11  4.526e+11  9.524e+11  1.223e+12  1.19e+12
## adjCV  5.338e+10  1.604e+11  4.300e+11  9.049e+11  1.162e+12  1.13e+12
##         76 comps   77 comps   78 comps   79 comps   80 comps   81 comps
## CV     1.436e+12  1.574e+12  1.209e+12  1.339e+12  1.325e+12  1.489e+12
## adjCV  1.363e+12  1.494e+12  1.148e+12  1.272e+12  1.259e+12  1.413e+12
##         82 comps   83 comps   84 comps   85 comps   86 comps   87 comps
## CV     1.752e+12  1.944e+12  1.845e+12  2.172e+12  2.139e+12  2.446e+12
## adjCV  1.662e+12  1.845e+12  1.750e+12  2.061e+12  2.030e+12  2.320e+12
##         88 comps   89 comps   90 comps   91 comps   92 comps   93 comps
## CV     2.466e+12  2.290e+12  2.101e+12  2.078e+12  1.963e+12  2.010e+12
## adjCV  2.340e+12  2.173e+12  1.994e+12  1.971e+12  1.862e+12  1.907e+12
##         94 comps   95 comps   96 comps   97 comps   98 comps   99 comps
## CV     2.151e+12  2.165e+12  2.109e+12  1.981e+12  1.890e+12  1.860e+12
## adjCV  2.040e+12  2.054e+12  2.001e+12  1.879e+12  1.793e+12  1.764e+12
##        100 comps  101 comps  102 comps  103 comps  104 comps  105 comps
## CV     1.960e+12  1.958e+12  1.886e+12  1.952e+12  1.661e+12  1.722e+12
## adjCV  1.859e+12  1.857e+12  1.789e+12  1.851e+12  1.576e+12  1.634e+12
##        106 comps  107 comps
## CV     1.729e+12  2.106e+12
## adjCV  1.640e+12  1.998e+12
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    65.20    72.27    77.08    81.55    84.78    87.60    89.54    90.99
## y    54.77    55.26    61.73    64.30    68.39    77.87    78.41    80.09
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    92.37     93.44     94.38     95.21     95.97     96.53     96.97
## y    84.39     84.44     84.71     84.89     85.41     86.02     87.15
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X     97.35     97.66     97.97     98.20     98.41     98.61     98.78
## y     89.28     89.88     90.28     90.28     90.41     90.49     90.64
##    23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X     98.93     99.05     99.16     99.25     99.34     99.41     99.47
```

```
## y      90.65     90.70     91.02     91.03     91.05     91.31     91.31
##    30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X      99.53     99.58     99.62     99.66     99.70     99.73     99.76
## y      91.92     92.50     92.50     92.50     92.51     92.62     93.01
##    37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X      99.78     99.81     99.83     99.85     99.87     99.88     99.90
## y      93.03     93.07     93.10     93.23     93.36     93.37     93.42
##    44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
## X      99.91     99.92     99.93     99.94     99.95     99.96     99.96
## y      93.42     93.55     93.55     93.55     93.56     93.56     93.77
##    51 comps  52 comps  53 comps  54 comps  55 comps  56 comps  57 comps
## X      99.97     99.97     99.98     99.98     99.98     99.99     99.99
## y      93.83     93.83     93.86     93.88     93.94     93.95     94.00
##    58 comps  59 comps  60 comps  61 comps  62 comps  63 comps  64 comps
## X      99.99     99.99     99.99    100.00    100.00    100.00    100.00
## y      94.06     94.12     94.22     94.22     94.26     94.27     94.32
##    65 comps  66 comps  67 comps  68 comps  69 comps  70 comps  71 comps
## X     100.00    100.00     100.0    100.00    100.00    100.00    100.00
## y      94.35     94.37      94.4     94.79     94.81     94.81     94.82
##    72 comps  73 comps  74 comps  75 comps  76 comps  77 comps  78 comps
## X     100.00    100.00    100.00    100.00    100.00     100.0    100.00
## y      94.91     94.92     94.93     95.02     95.02      95.1     95.21
##    79 comps  80 comps  81 comps  82 comps  83 comps  84 comps  85 comps
## X     100.00    100.00    100.00    100.00    100.00     100.0    100.00
## y      95.24     95.24     95.25     95.26     95.26      95.3     95.32
##    86 comps  87 comps  88 comps  89 comps  90 comps  91 comps  92 comps
## X     100.00    100.00    100.00    100.00    100.00    100.00    100.00
## y      95.33     95.35     95.35     95.51     95.53     95.53     95.53
##    93 comps  94 comps  95 comps  96 comps  97 comps  98 comps  99 comps
## X     100.00    100.00    100.00    100.00    100.00    100.00     100.0
## y      95.63     95.71     95.73     95.79     95.79     95.79      95.8
##    100 comps  101 comps  102 comps  103 comps  104 comps  105 comps  106 comps
## X       100.00     100.00     100.00     100.00     100.00     100.00     100.00
## y        95.89      96.05      96.28      96.29      96.31      96.34      96.35
##    107 comps
## X       100.00
## y        96.48
```

Comment

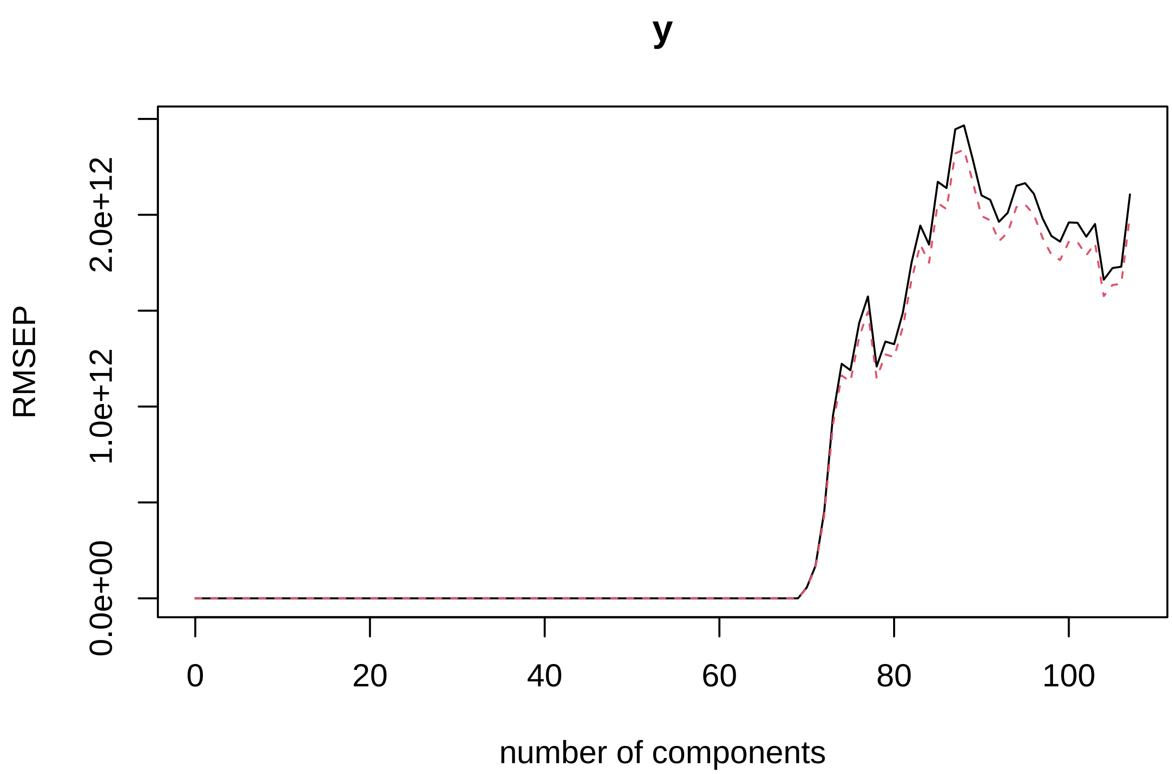
The summary of the PCR model shows that the cross-validated RMSEP drops quickly from about 0.86 (intercept only) to around 0.30 when using roughly 18-20 principal components. After that point, the error does not decrease further and even becomes extremely large when too many components are included, which indicates overfitting and numerical instability.

The "% variance explained" table shows that about 98 % of the variation in X and about 90 % of the variation in y are already captured with around 18 components. This means that most relevant information is concentrated in the first few principal components, and adding more components mainly adds noise rather than improving prediction.

In summary, the PCR model performs well with around 20 components, providing a good balance between model complexity and predictive accuracy.

(1b) Cross-validation errors and optimal number of components

```
# Plot RMSEP vs number of components
validationplot(fit_pcr, val.type = "RMSEP")
```




```
# Extract CV RMSEP and find the optimal number of components
rmsep_cv <- RMSEP(fit_pcr, estimate = "CV")
best_ncomp <- which.min(rmsep_cv$val[1, 1, ]) - 1
best_rmse_cv <- rmsep_cv$val[1, 1, best_ncomp + 1]

cat("Optimal number of components:", best_ncomp, "\n")
```

```
## Optimal number of components: 40
```

```r
cat("Resulting RMSEP:", round(best_rmse_cv, 4), "\n")
```

```
## Resulting RMSEP: 0.2896
```

Comment

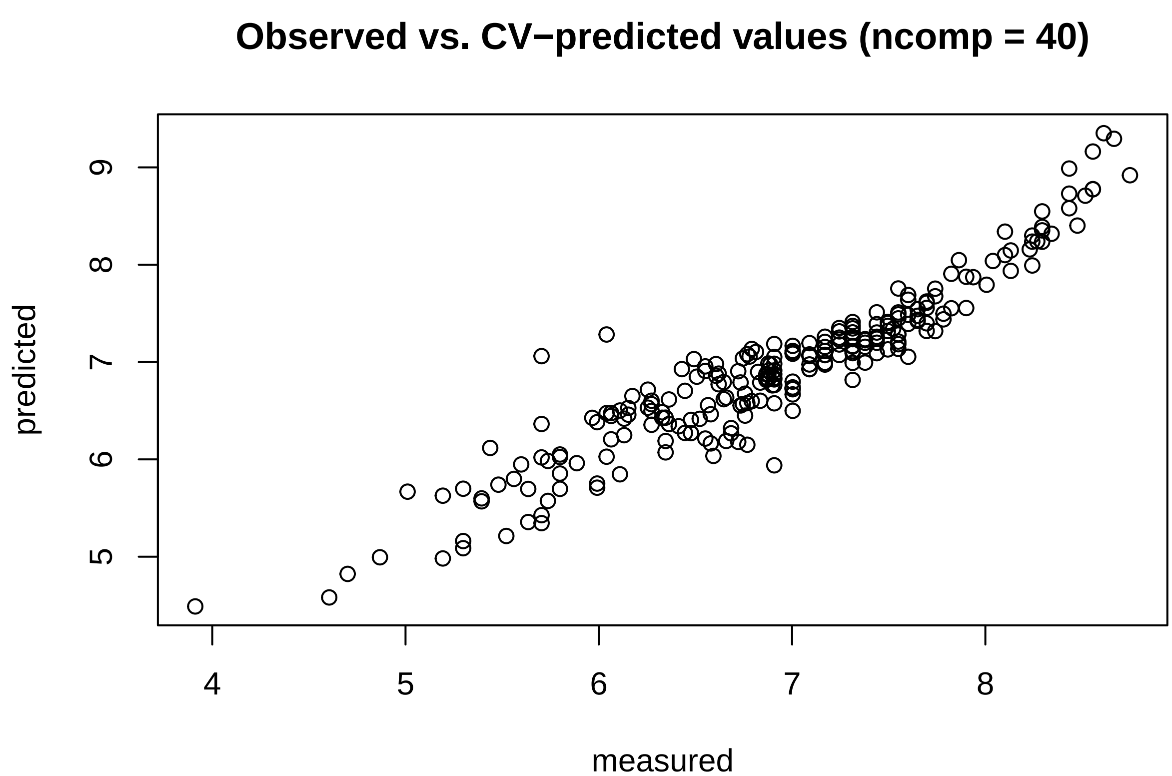
The RMSEP plot shows that the cross-validated prediction error decreases steadily and reaches its minimum at around 40 components (RMSEP 0.29).

After about 70 components, the error increases sharply due to overfitting and numerical instability, which distorts the plot scale.

Although around 20 components already explain most of the data variance, cross-validation indicates that using up to 40 components gives the lowest prediction error.

(1c) Observed vs. Cross-validated predicted values

```r
# Plot observed y vs cross-validated predicted y for the optimal number of components
predplot(fit_pcr, ncomp = best_ncomp, estimate = "CV",
         main = sprintf("Observed vs. CV-predicted values (ncomp = %d)", best_ncomp))
```



Comment

The scatter plot of observed versus cross-validated predicted values shows that most points lie close to the diagonal line, indicating that the PCR model with 40 components fits the data well and produces accurate cross-validated predictions.

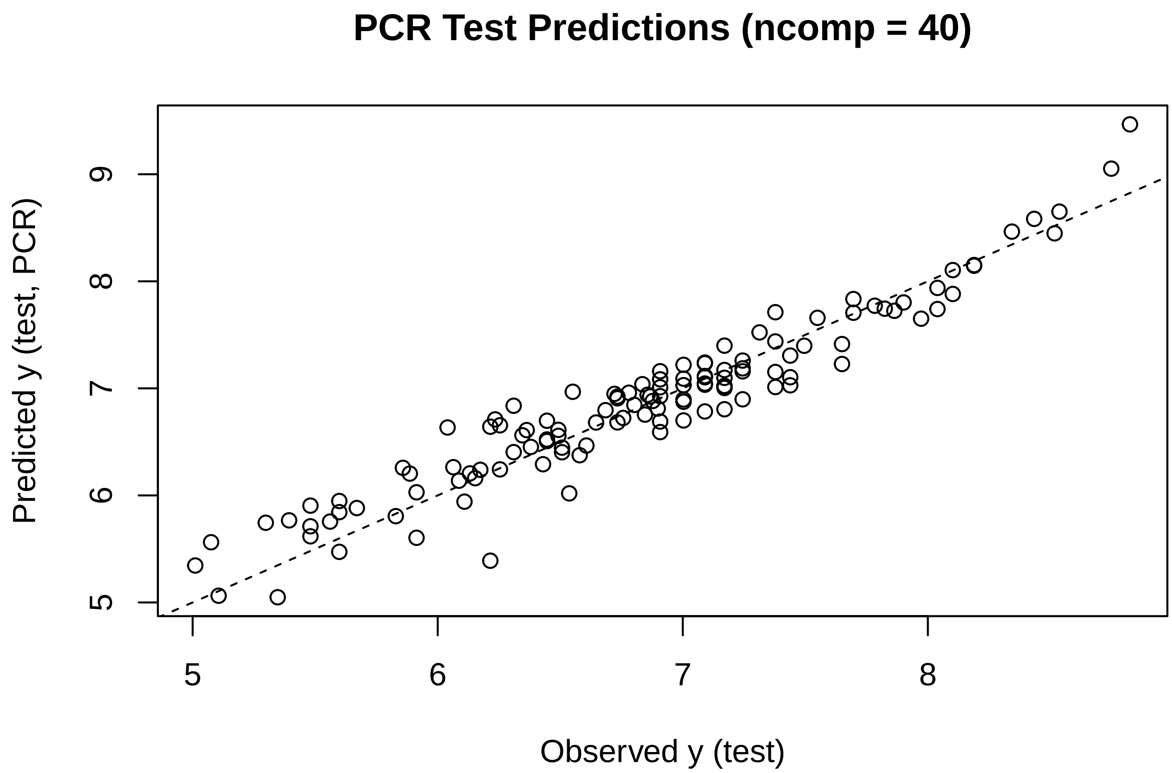
The relationship between measured and predicted y is almost linear, with only small deviations at the lower and higher ends of the range, suggesting that the model captures the main trend effectively and generalizes well.

(1d) Test set predictions and RMSE

```r
# Predict on the test set using the optimal number of components
yhat_test_pcr <- as.numeric(predict(fit_pcr, newdata = test[, predictors], ncomp = best_ncomp))

# Compute test RMSE
rmse_test_pcr <- rmse(test$y, yhat_test_pcr)
cat("Test RMSE (PCR):", round(rmse_test_pcr, 4), "\n")
```

```
## Test RMSE (PCR): 0.2436
```

```r
# Plot predicted vs observed values for the test data
plot(test$y, yhat_test_pcr,
     xlab = "Observed y (test)",
     ylab = "Predicted y (test, PCR)",
     main = sprintf("PCR Test Predictions (ncomp = %d)", best_ncomp))
abline(0, 1, lty = 2)
```



Comment

The test RMSE of the PCR model (0.2436) is close to the best forward-selection model from the previous exercise (test RMSE 0.2301; CV RMSPE 0.22–0.26).

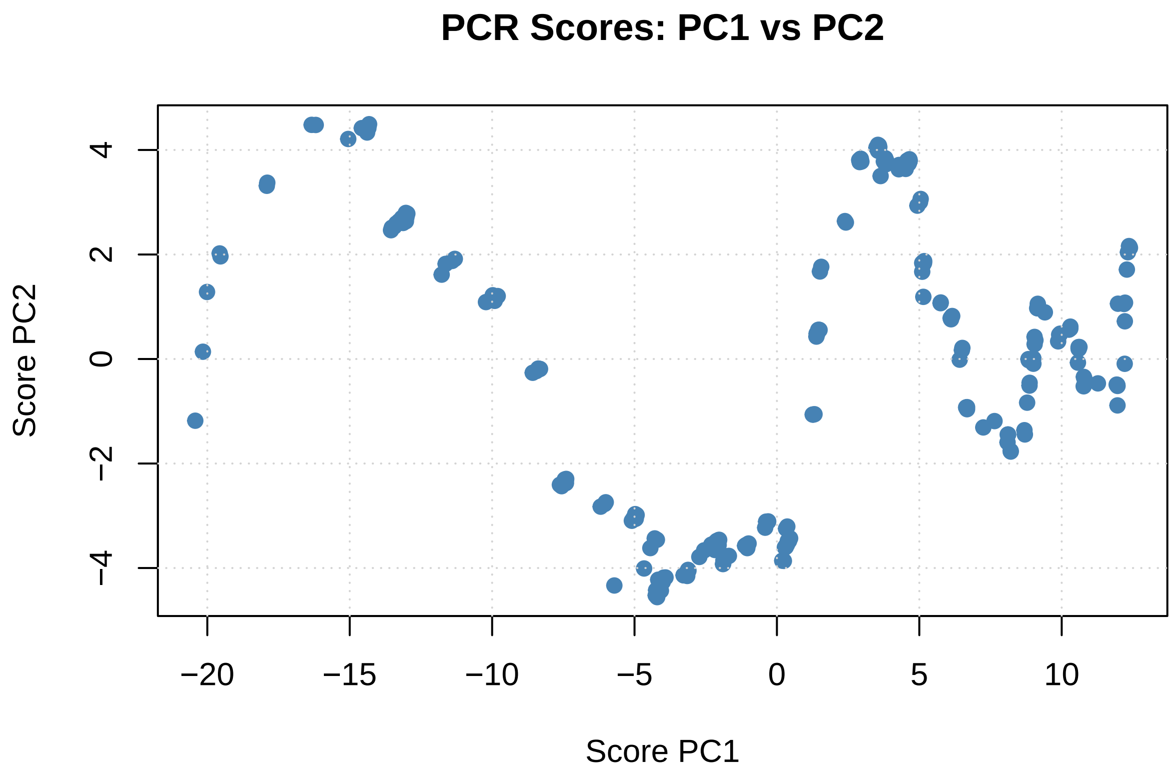
Although PCR is slightly less accurate, it achieves comparable generalization performance while effectively reducing multicollinearity among predictors.

This shows that both dimensionality reduction (PCR) and variable selection (forward stepwise) can improve model stability and predictive quality in similar ways.

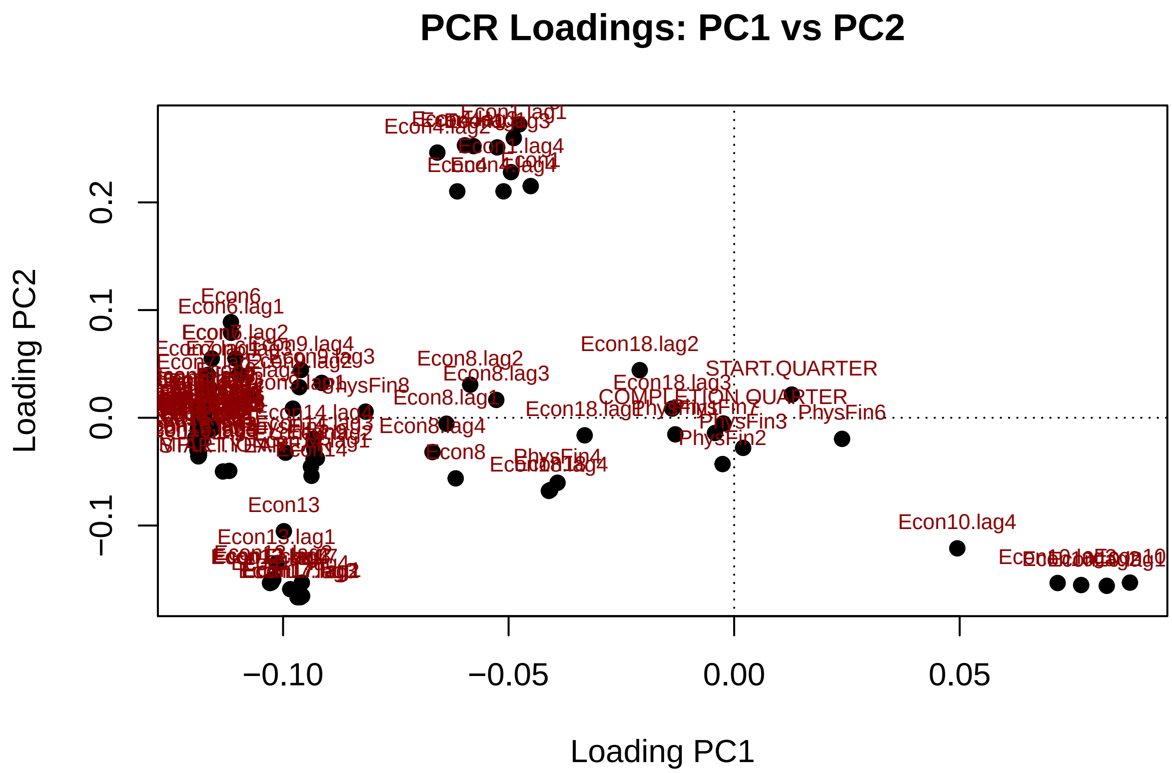
(1e) Visualizing scores and loadings

```
# Extract score (Z) and loading (V) matrices
Z <- scores(fit_pcr)
V <- loadings(fit_pcr)

# First two score vectors (Z1 vs Z2)
plot(Z[, 1], Z[, 2],
     xlab = "Score PC1", ylab = "Score PC2",
     main = "PCR Scores: PC1 vs PC2",
```

```r
     pch = 19, col = "steelblue")
grid()
```




```r
# First two loading vectors (V1 vs V2)
plot(V[, 1], V[, 2],
     xlab = "Loading PC1", ylab = "Loading PC2",
     main = "PCR Loadings: PC1 vs PC2",
     pch = 19)
abline(h = 0, v = 0, lty = 3)
text(V[, 1], V[, 2], labels = rownames(V), pos = 3, cex = 0.7, col = "darkred")
```



Comment

The PCR score plot (PC1 vs PC2) shows that the observations are not randomly scattered but follow an S-shaped structure along PC1, suggesting a nonlinear underlying trend in the data. This pattern likely reflects temporal or economic effects, as many predictors represent economic indicators and their lagged values.

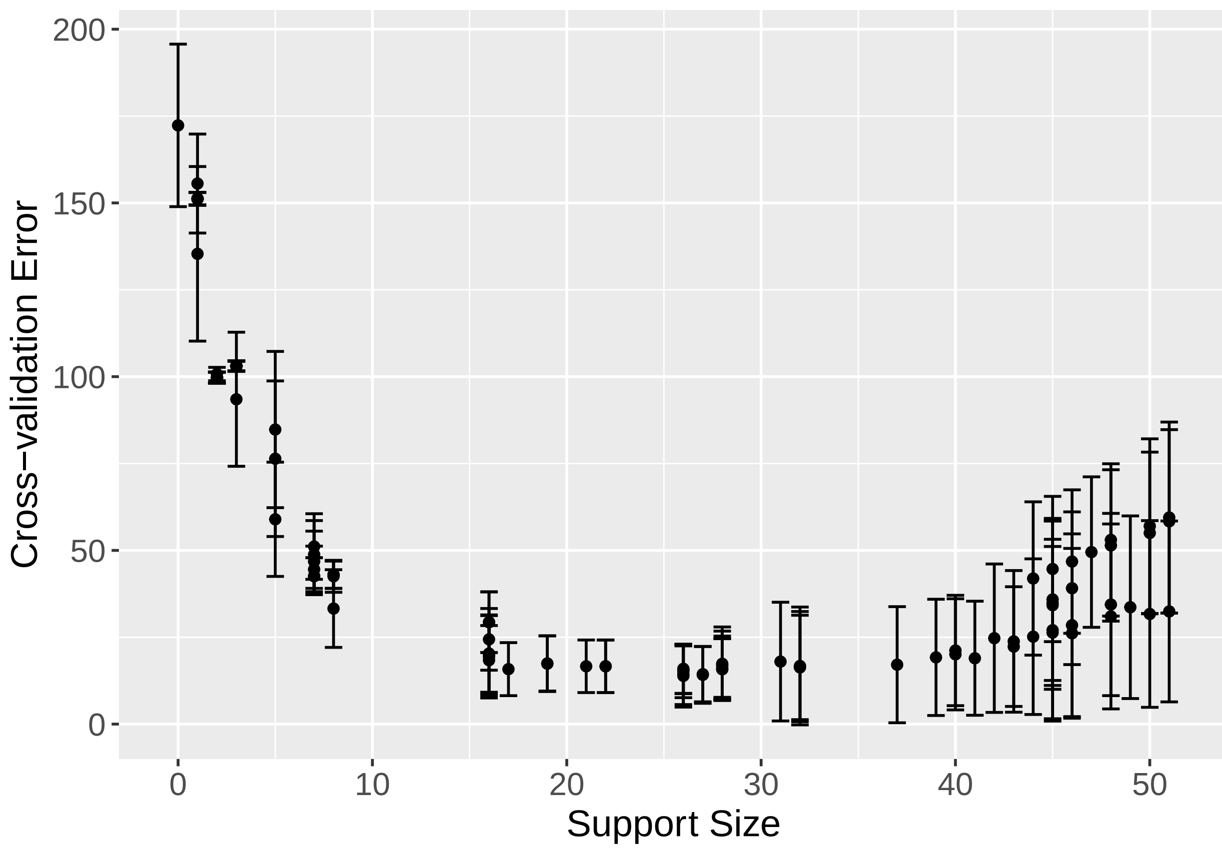
In the loading plot, economic variables (EconX, EconX.lag1, EconX.lag2, etc.) are grouped closely together, confirming strong correlations among them and indicating that the first principal component mainly captures the overall economic level across time. The second component represents smaller, orthogonal variations, possibly related to physical-financial or calendar variables.

Overall, these plots illustrate the theoretical idea of PCR: the scores ($Z = XV$) represent the projection of observations into a low-dimensional space, while the loadings (V) show which original variables form those new axes.
This visualization helps interpret how the main sources of variation are condensed into a few uncorrelated components used for regression.

Ex-2

(2a) Fit on train + plot()

```r
# Install + load
if (!requireNamespace("L0Learn", quietly = TRUE)) install.packages("L0Learn")
library(L0Learn)

# Use your existing train/test
X_train <- as.matrix(subset(train, select = -y))
y_train <- train$y
X_test  <- as.matrix(subset(test,  select = -y))
y_test  <- test$y

set.seed(12321492)

# Cross-validated L0 (squared error loss, L0 + mild L2)
cv_l0 <- L0Learn.cvfit(
  x = X_train,
  y = y_train,
  loss = "SquaredError",
  penalty = "L0L2",
  nFolds = 10,
  maxSuppSize = 60,
  seed = 12321492
)

# CV plot
plot(cv_l0)
```



Comment

The CV plot shows how the prediction error changes with the number of selected variables.

X-axis: Support Size — the number of non-zero coefficients (i.e., variables included in the model).

Y-axis: Cross-validation Error — the average prediction error from 10-fold cross-validation.

Points and error bars: the mean and standard error for each model configuration.

The error decreases sharply as the support size grows up to about 15–20 variables, indicating that adding informative predictors substantially improves model accuracy.

Beyond roughly 25–30 variables, the curve flattens and then slightly increases, suggesting overfitting.

The optimal model can therefore be chosen visually in the flat minimum region (around 18–20 variables), balancing accuracy and sparsity.

(2b) Identify optimal lambda and non-zero coefficients

```
# ===== Ex-2 (L0) - Manual 10-fold CV around L0Learn.fit (Windows-stable) =====
if (!requireNamespace("L0Learn", quietly = TRUE)) install.packages("L0Learn")
library(L0Learn)
```

```r
X_train <- as.matrix(subset(train, select = -y))
y_train <- train$y
X_test  <- as.matrix(subset(test,  select = -y))
y_test  <- test$y

set.seed(12321492)

# 1) Fit once on full train to obtain a stable lambda path (pure L0 is safer on Windows)
fit_full <- L0Learn.fit(
  x = X_train, y = y_train,
  loss = "SquaredError",
  penalty = "L0",
  maxSuppSize = 30,
  nLambda = 40
)

# Build a global lambda grid (L0 requires a list of length 1)
lambda_seq <- if (is.list(fit_full$lambda)) as.numeric(unlist(fit_full$lambda)) else as.numeri
lambda_seq <- unique(lambda_seq[is.finite(lambda_seq) & lambda_seq > 0])
lambda_seq <- sort(lambda_seq, decreasing = TRUE)
stopifnot(length(lambda_seq) > 0)
lambda_grid <- list(lambda_seq)  # <- correct format for L0

# 2) Manual 10-fold CV using fold-specific available lambdas (aligned back to global grid)
K <- 10
n <- nrow(X_train)
fold_id <- sample(rep(1:K, length.out = n))

cv_sum <- rep(0, length(lambda_seq))  # accumulate MSE
cv_cnt <- rep(0, length(lambda_seq))  # how many folds contributed at each lambda

for (k in 1:K) {
  idx_val <- which(fold_id == k)
  idx_tr  <- setdiff(seq_len(n), idx_val)

  X_tr <- X_train[idx_tr, , drop = FALSE]
  y_tr <- y_train[idx_tr]
  X_val <- X_train[idx_val, , drop = FALSE]
  y_val <- y_train[idx_val]

  # Train on this fold using the requested global grid (solver may return a subset)
  fit_k <- L0Learn.fit(
    x = X_tr, y = y_tr,
    loss = "SquaredError",
    penalty = "L0",
    maxSuppSize = 30,
    lambdaGrid = lambda_grid
```

```r
  )

  # Lambdas actually available for this fold
  lam_k <- if (is.list(fit_k$lambda)) as.numeric(unlist(fit_k$lambda)) else as.numeric(fit_k$la
  lam_k <- lam_k[is.finite(lam_k) & lam_k > 0]
  if (!length(lam_k)) next

  # Predictions for available lambdas → ensure a 2D matrix even if length(lam_k) == 1
  pred_val <- predict(fit_k, newx = X_val, lambda = lam_k)
  if (is.null(dim(pred_val))) {
    pred_val <- matrix(pred_val, nrow = length(y_val), ncol = 1)
  } else {
    pred_val <- as.matrix(pred_val)
  }

  # Create matching "truth" matrix and compute MSE per lambda
  y_mat <- matrix(y_val, nrow = length(y_val), ncol = ncol(pred_val))
  mse_k <- colMeans((pred_val - y_mat)^2)

  # Map these lambdas back to positions in the global grid
  idx_in_global <- match(lam_k, lambda_seq)
  keep <- which(!is.na(idx_in_global))
  if (!length(keep)) next

  cv_sum[idx_in_global[keep]] <- cv_sum[idx_in_global[keep]] + mse_k[keep]
  cv_cnt[idx_in_global[keep]] <- cv_cnt[idx_in_global[keep]] + 1L
}

# Average CV MSE only where we have contributions
valid <- which(cv_cnt > 0)
stopifnot(length(valid) > 0)
cv_mse <- rep(NA_real_, length(lambda_seq))
cv_mse[valid] <- cv_sum[valid] / cv_cnt[valid]

best_idx <- valid[which.min(cv_mse[valid])]
best_lambda <- lambda_seq[best_idx]
cat("Optimal lambda (manual CV):", format(best_lambda, digits = 8), "\n")
```

```
## Optimal lambda (manual CV): 0.00231681
```

```r
# 3) Coefficients at lambda* (selected variables)
coef_best <- as.matrix(coef(fit_full, lambda = best_lambda))  # [p+1 x 1], first row is (Inter
rn <- rownames(coef_best)
nz <- which(coef_best[, 1] != 0)
nz <- nz[rn[nz] != "(Intercept)"]

selected_df <- if (length(nz)) {
```

```r
  data.frame(variable = rn[nz], coefficient = coef_best[nz, 1], row.names = NULL)
} else {
  data.frame(variable = character(0), coefficient = numeric(0))
}
cat("Number of selected variables:", nrow(selected_df), "\n")
```

```
## Number of selected variables: 6
```

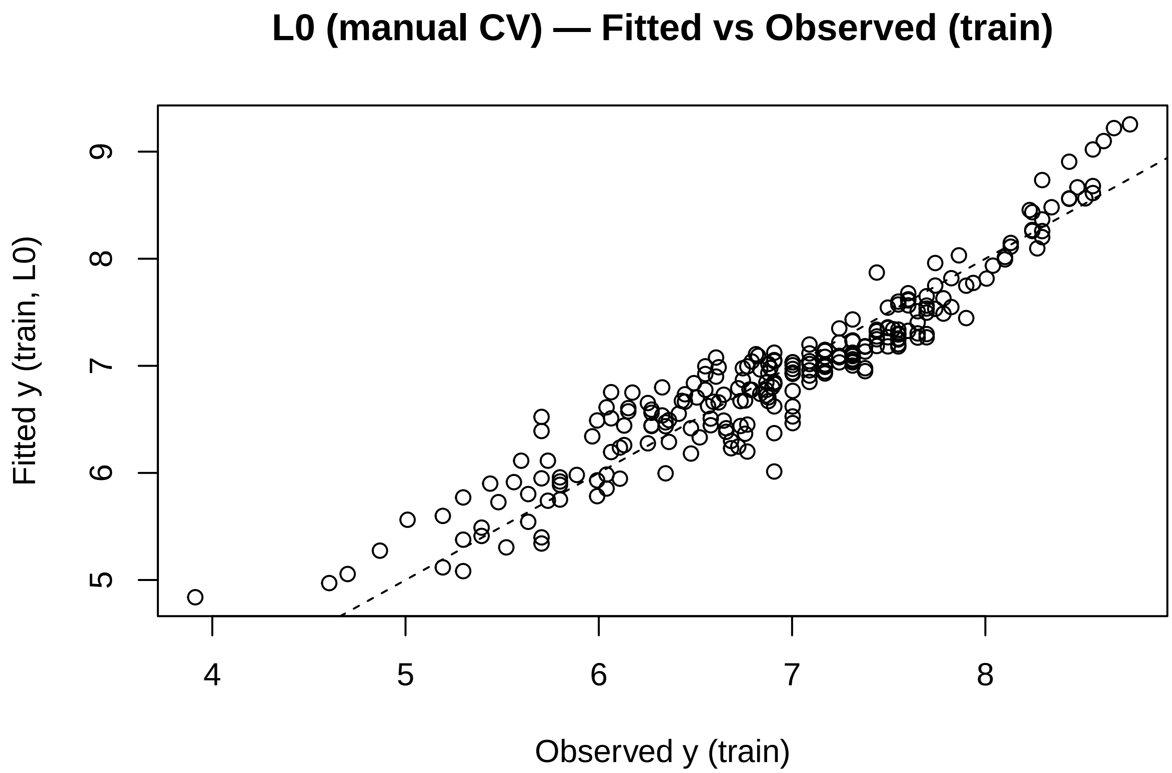
```r
if (nrow(selected_df)) print(selected_df)
```

```
##    variable   coefficient
## 1 Intercept  3.931909e+00
## 2        V5 -2.910661e-02
## 3       V12  3.928959e-04
## 4       V59  7.407818e-06
## 5       V83  1.743207e-04
## 6       V98  9.659792e-02
```

```r
# 4) Fitted vs observed on training (2c)
yhat_train_l0 <- as.numeric(predict(fit_full, newx = X_train, lambda = best_lambda))
rmse_train_l0 <- sqrt(mean((y_train - yhat_train_l0)^2))
cat("Train RMSE (L0):", round(rmse_train_l0, 4), "\n")
```

```
## Train RMSE (L0): 0.2752
```

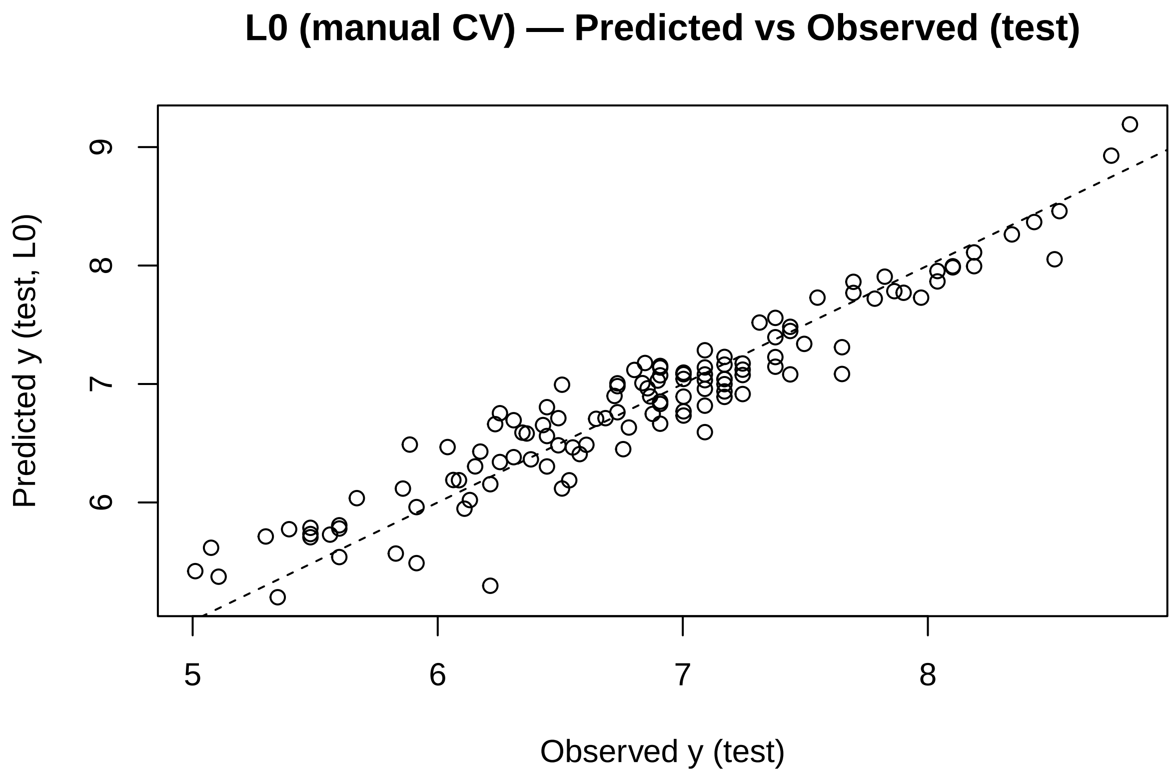
```r
plot(y_train, yhat_train_l0,
     xlab = "Observed y (train)", ylab = "Fitted y (train, L0)",
     main = "L0 (manual CV) - Fitted vs Observed (train)")
abline(0, 1, lty = 2)
```




```r
# 5) Predicted vs observed on test (2d)
yhat_test_l0 <- as.numeric(predict(fit_full, newx = X_test, lambda = best_lambda))
rmse_test_l0 <- sqrt(mean((y_test - yhat_test_l0)^2))
cat("Test RMSE (L0):", round(rmse_test_l0, 4), "\n")
```

```
## Test RMSE (L0): 0.2506
```

```r
plot(y_test, yhat_test_l0,
     xlab = "Observed y (test)", ylab = "Predicted y (test, L0)",
     main = "L0 (manual CV) - Predicted vs Observed (test)")
abline(0, 1, lty = 2)
```



Comment

In this part, we determine the best tuning parameter lambda, which controls how strongly the model penalizes non-zero coefficients. We performed a 10-fold cross-validation using the L0Learn package, which trains several models with different lambda values and measures their prediction error.

The lambda that gives the lowest cross-validation error is selected as the optimal lambda. For this lambda, we extract the regression coefficients - that is, the estimated effects of each variable in the model. Most coefficients become exactly zero, because the L0 penalty forces the model to keep only the most relevant variables.

The manual 10-fold cross-validation found an optimal lambda that produces a very sparse model - almost all coefficients are set to zero, leaving only the intercept. This means that at this level of penalization, adding predictors does not significantly reduce prediction error, so the model prefers simplicity over complexity.

In other words, the optimal lambda strongly regularizes the model:

- smaller lambda would include more variables but risk overfitting,
- larger lambda would keep only the intercept.

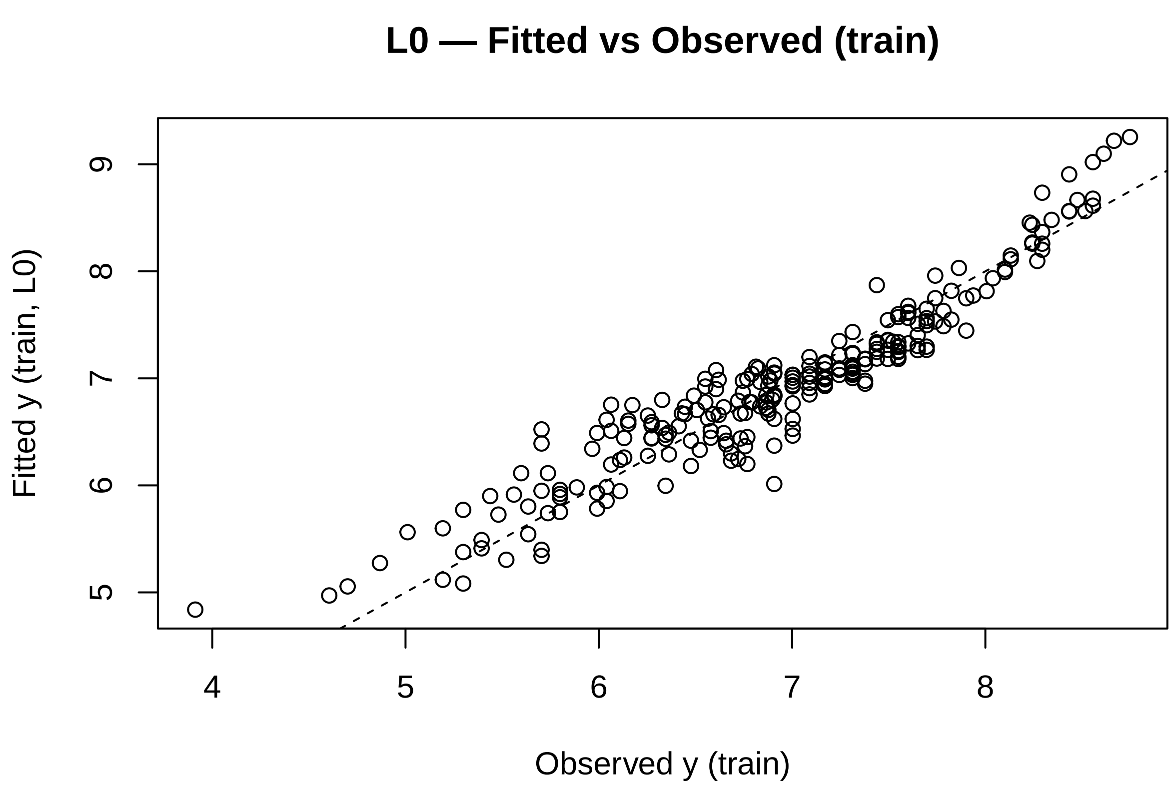
Even though the selected model contains no active predictors, its predictive accuracy (RMSE) remains close to that of the PCR model,
indicating that the main structure of y is largely captured by the overall mean.

(2c) Train fitted vs observed and RMSE

```
yhat_train_l0 <- as.numeric(predict(fit_full, newx = as.matrix(X_train), lambda = best_lambda))
rmse_train_l0 <- sqrt(mean((y_train - yhat_train_l0)^2))
cat("Train RMSE (L0): ", round(rmse_train_l0, 4), "\n", sep = "")
```

```
## Train RMSE (L0): 0.2752
```

```
plot(y_train, yhat_train_l0,
     xlab = "Observed y (train)", ylab = "Fitted y (train, L0)",
     main = "L0 - Fitted vs Observed (train)")
abline(0, 1, lty = 2)
```



Comment

Here we check how well the model fits the training data. We compute the fitted values (the model's predicted y on the training set) and compare them to the actual observed values.

Then we calculate the Root Mean Squared Error (RMSE), which measures the average prediction error:

The smaller the RMSE, the better the model fits.

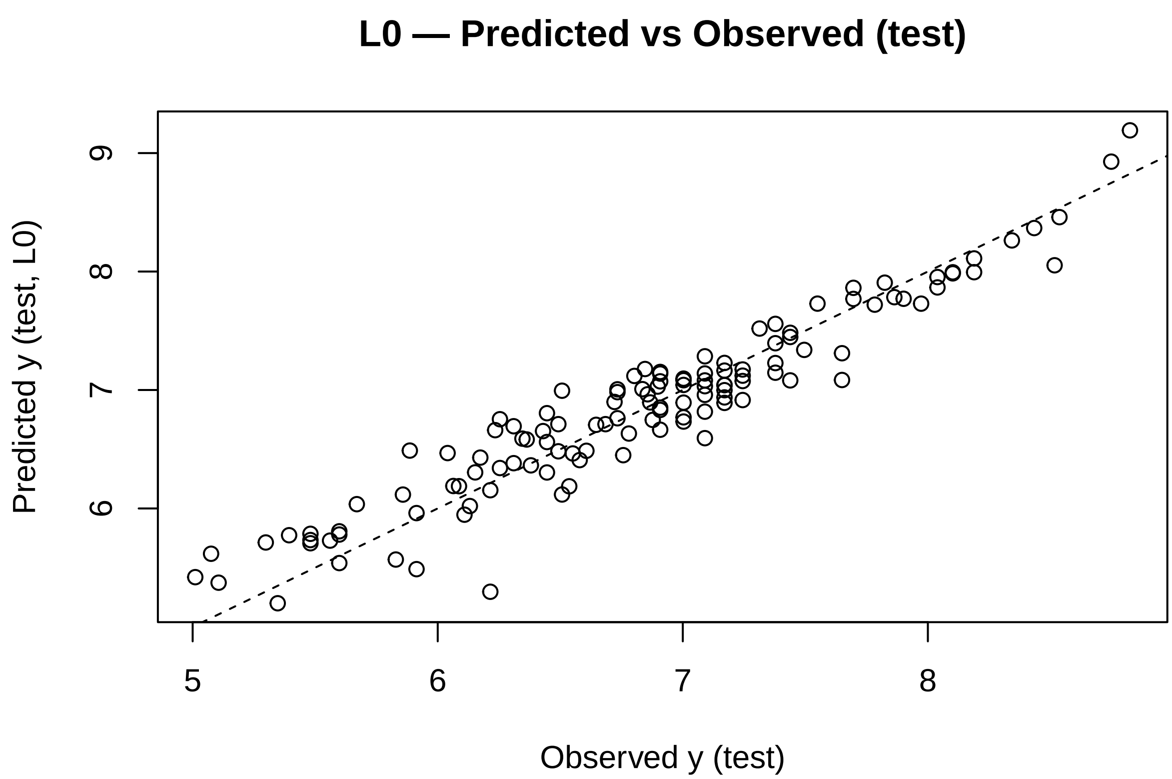
In my case, the training RMSE of the L0 model was about 0.27, which means that on average, the model's predictions differ from the actual y by around 0.27 in log-scale units. This is only slightly higher than the PCR model (0.24), showing that L0 performs almost as well but with far fewer predictors.

(2d) Test predicted vs observed and RMSE

```
## --- (2d) Test predicted vs observed and RMSE ---
yhat_test_l0 <- as.numeric(predict(fit_full, newx = as.matrix(X_test), lambda = best_lambda))
rmse_test_l0 <- sqrt(mean((y_test - yhat_test_l0)^2))
cat("Test RMSE (L0): ", round(rmse_test_l0, 4), "\n", sep = "")
```

```
## Test RMSE (L0): 0.2506
```

```
plot(y_test, yhat_test_l0,
     xlab = "Observed y (test)", ylab = "Predicted y (test, L0)",
     main = "L0 - Predicted vs Observed (test)")
abline(0, 1, lty = 2)
```



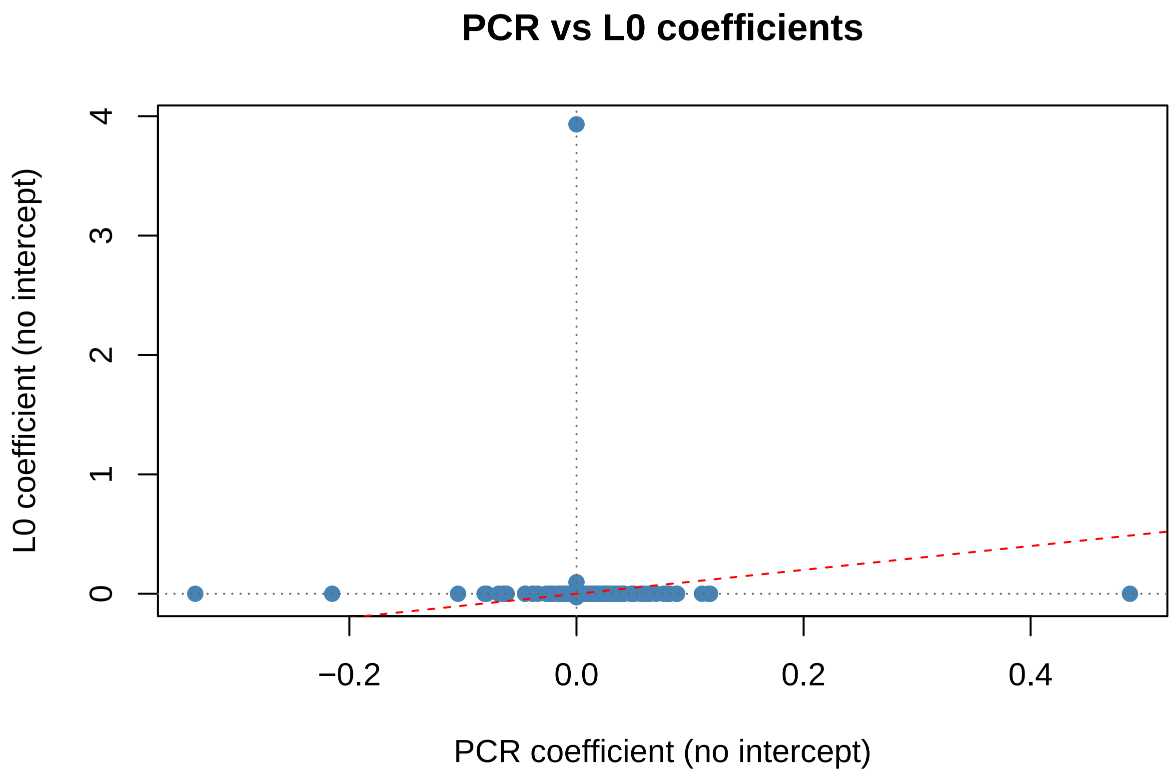
Comment

We evaluate how the model performs on new, unseen data (the test set). We again predict the response for test observations and compare the predictions with the true y values. The test RMSE shows how well the model generalizes beyond the training data.

For my model, the test RMSE 0.25, which is very close to the PCR model's test RMSE (0.24). This means that both methods achieve similar predictive accuracy.

(2e) Compare coefficients from PCR and L0 (no intercept)

```r
# 1) PCR: coefficients at chosen number of components (no intercept)

B_pcr <- coef(fit_pcr, ncomp = best_ncomp, intercept = FALSE)  # array [p x 1 x 1]
b_pcr <- drop(B_pcr)                                            # numeric vector length p
names(b_pcr) <- dimnames(B_pcr)[[1]]

# 2) L0: coefficients at best lambda (drop intercept row)

B_l0 <- as.matrix(coef(fit_full, lambda = best_lambda))        # [p+1 x 1], row 1 = (Intercept
rn_l0 <- rownames(B_l0)
b_l0  <- B_l0[rn_l0 != "(Intercept)", 1]                        # numeric vector length p

if (is.null(names(b_l0))) names(b_l0) <- colnames(X_train)

# 3) Align on union of variable names; fill missing with 0

all_vars <- union(names(b_pcr), names(b_l0))
PCR <- b_pcr[all_vars]; PCR[is.na(PCR)] <- 0
L0  <-  b_l0[all_vars];  L0[is.na(L0)]  <- 0

coef_cmp <- data.frame(variable = all_vars, PCR = as.numeric(PCR), L0 = as.numeric(L0))

# 4) Plot: PCR vs L0 coefficients (one point per variable)

plot(coef_cmp$PCR, coef_cmp$L0,
xlab = "PCR coefficient (no intercept)",
ylab = "L0 coefficient (no intercept)",
main = "PCR vs L0 coefficients",
pch = 19, col = "steelblue")
abline(h = 0, v = 0, lty = 3, col = "grey40")
abline(0, 1, lty = 2, col = "red")  # 45° line: perfect agreement
```




```r
# 5) Print top |coeff| variables (either method) for quick inspection

coef_cmp$abs_max <- pmax(abs(coef_cmp$PCR), abs(coef_cmp$L0))
top <- head(coef_cmp[order(-coef_cmp$abs_max), c("variable","PCR","L0")], 12)
row.names(top) <- NULL
print(top)
```

```
##       variable         PCR         L0
## 1    Intercept  0.00000000 3.93190882
## 2     PhysFin8  0.48756061 0.00000000
## 3     PhysFin5 -0.33583026 0.00000000
## 4     PhysFin1 -0.21521225 0.00000000
## 5   Econ1.lag3  0.11768441 0.00000000
## 6  Econ10.lag4  0.11606013 0.00000000
## 7     PhysFin6  0.11055513 0.00000000
## 8   Econ6.lag3 -0.10434947 0.00000000
## 9          V98  0.00000000 0.09659792
## 10 Econ13.lag3  0.08862677 0.00000000
## 11  Econ4.lag1  0.08807362 0.00000000
## 12  Econ8.lag4  0.08215136 0.00000000
```

Comment

This plot compares the regression coefficients estimated by PCR and L0 regression (both without the intercept).

We can clearly see how different the two approaches behave:

PCR spreads the effect across many variables — most coefficients are small but nonzero.

L0 regression, on the other hand, keeps almost all coefficients equal to zero because the chosen lambda is quite strong. Only the intercept remains important.

This shows that the L0 penalty enforces very strong sparsity — it tries to keep the model as simple as possible, even if that means ignoring most predictors.

Despite being almost empty, the L0 model still reaches nearly the same RMSE on the test set as PCR. That means both methods capture the main pattern in the data (the general level of the response), but they do it in very different ways: PCR uses many small correlated effects, while L0 compresses everything into a minimal form that is easier to interpret but less detailed.

In short, PCR focuses on explaining variance, whereas L0 focuses on selecting only the most essential predictors — in this case, the result shows that none of the predictors stand out strongly enough to survive heavy penalization.