Tarea 1

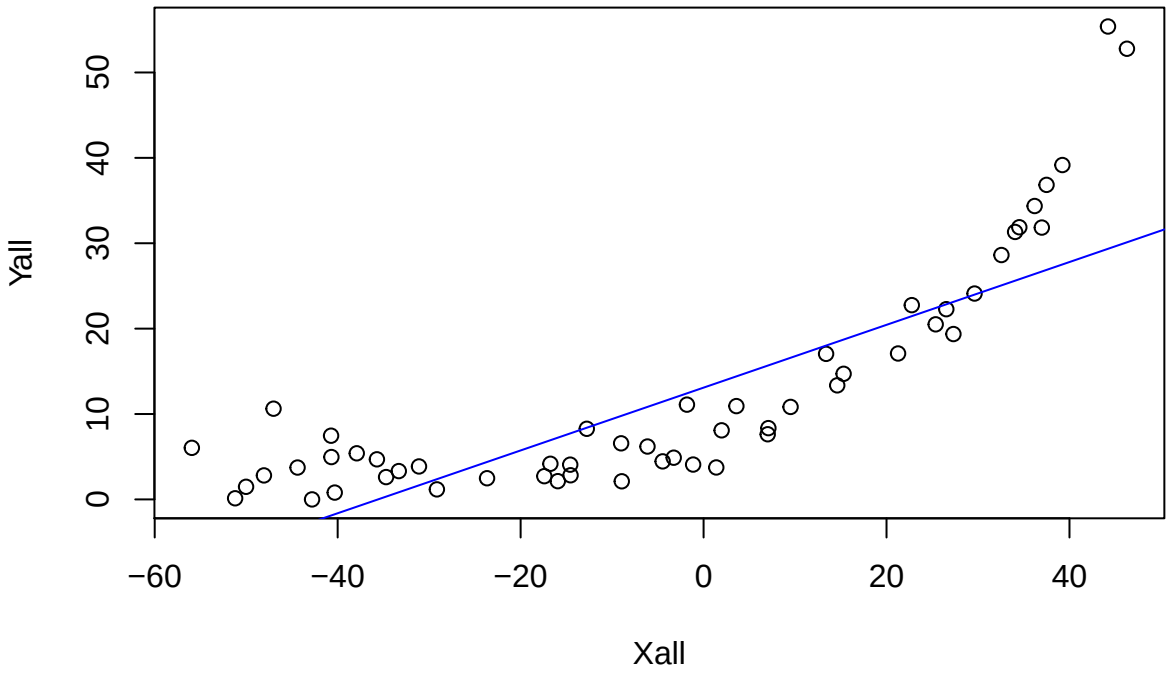
Maximiliano Norbu

5/4/2022

Parte 1

Pregunta 1

```
modelo = lm(Yt ~ Xt)
plot(Xall, Yall)
abline(modelo, col="BLUE")
```



Hay mucho sesgo pues los datos reales se alejan mucho de mis predicciones (se nota visualmente que los datos no están cerca de la curva que ajustó mi regresión lineal). Varianza no se nota demasiada al ojo.

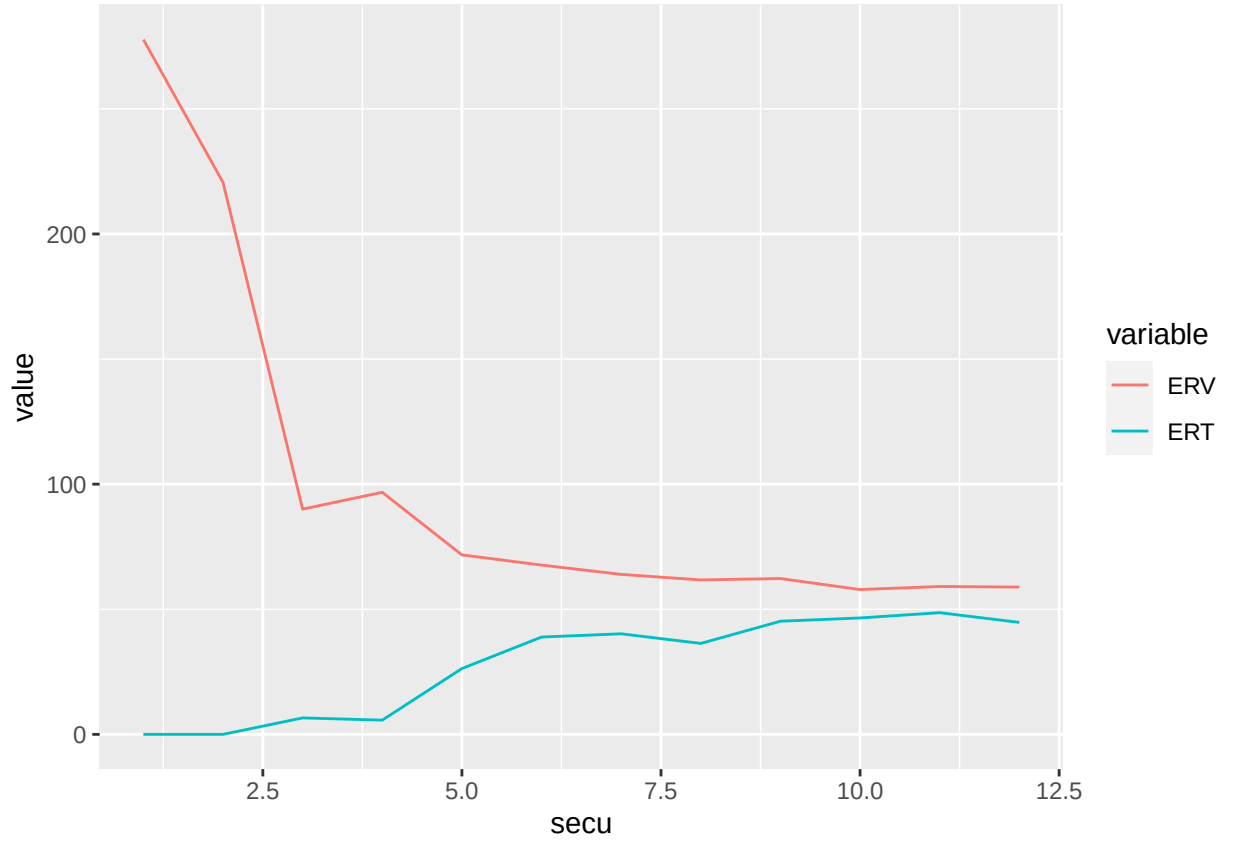
Pregunta 2

```
modelos = seq(1:12)
for (i in 2:12){
  modelos[i] = lm(Yt[1:i]~ Xt[1:i])
}
```

```
## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo

## Warning in modelos[i] <- lm(Yt[1:i] ~ Xt[1:i]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo
```

```
modelos[[1]][[1]] = Yt[1]
ERT = seq(1:12)
ERV = seq(1:12)

ERT[1] = 0
ERV[1] = 0
for (j in 1:21){
  ERV[1] = ERV[1] + (Yv[j]-(modelos[[1]][[1]]))^2
}
ERV[1]= ERV[1]/21

for (i in 2:12){
```

```
  rata = 0
  for (j in 1:i){
    rata = rata+ (Yt[j]-(modelos[[i]][[1]]+Xt[j]*modelos[[i]][[2]]))^2
  }
  ERT[i] = rata/i
}

for (i in 2:12){
  rata = 0
  for (j in 1:21){
    rata = rata+ (Yv[j]-(modelos[[i]][[1]]+Xv[j]*modelos[[i]][[2]]))^2
  }
  ERV[i] = rata/21
}
secu = seq(1:12)
df = data.frame(secu, ERV, ERT)
df2 = melt(data = df, id.vars = "secu")
ggplot(data = df2, aes(x = secu, y = value, colour = variable)) + geom_line()
```



No aprecio problemas de varianza, ya que, tanto los errores de la base de testeo como los errores de la base de validación, parecieran converger a un valor. Sin embargo, noto problemas de sesgo, pues el error cuadrático medio al que se converge es cercano al 50 y los datos que tenemos fluctuan, aproximadamente entre -50 y 50, por lo que el error es demasiado grande.

Pregunta 3

```
df8 = poly(Xall, degree = 8, raw=T)
medias = apply(df8,2,mean)
varianzas = apply(df8,2,var)
df8N = scale(df8, center = T, scale = T)
df8N
```

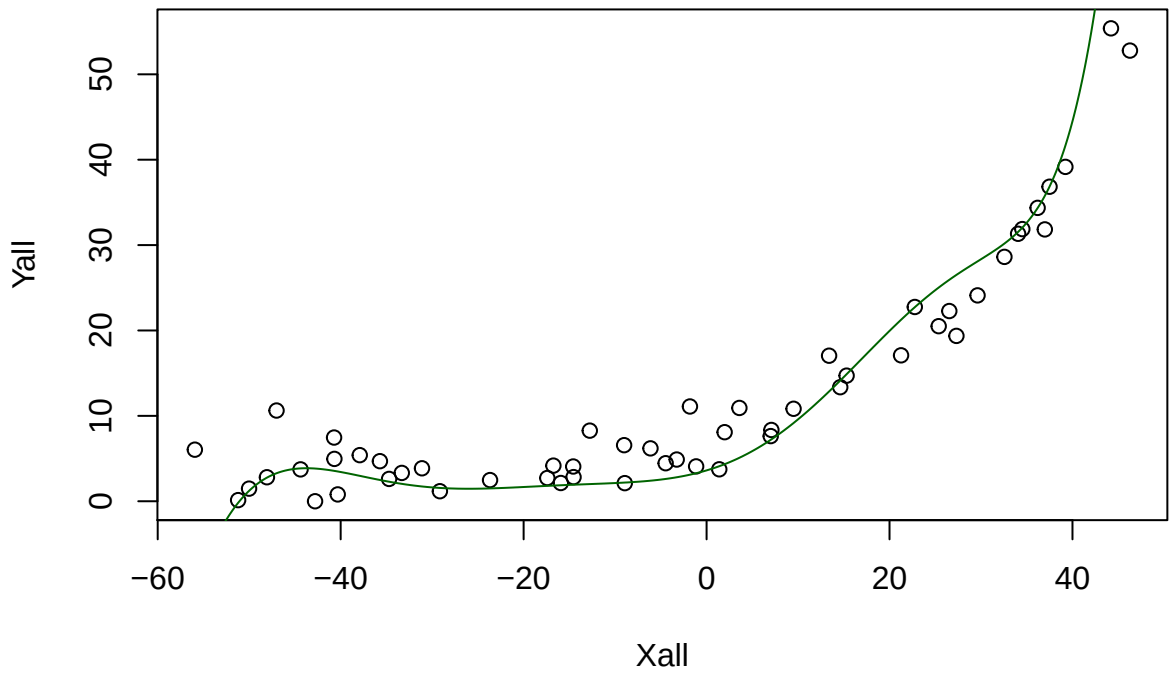
```
##                 1           2           3           4           5           6
##  [1,] -0.381170822 -0.77292950  0.15344300 -0.67741811  0.25736912 -0.522600956
##  [2,] -0.823248336 -0.05753992 -0.23442096 -0.36418039  0.09316524 -0.416284617
##  [3,]  1.362434099  0.49442887  1.11601174  0.10863838  0.77468354 -0.125802454
##  [4,]  1.406006863  0.60965247  1.21526464  0.23274011  0.87310508 -0.031325327
##  [5,] -1.455641758  1.69487746 -1.84770830  1.83202716 -1.83590601  1.666728697
##  [6,] -0.147180720 -0.98185104  0.21580161 -0.70509068  0.26532821 -0.525425103
##  [7,]  0.663947398 -0.79652117  0.29629409 -0.68198756  0.27268752 -0.523226612
##  [8,] -1.009003638  0.36818736 -0.55302062 -0.01725834 -0.14700811 -0.214561254
##  [9,]  0.198374290 -1.07551297  0.22922752 -0.70813261  0.26579679 -0.525516031
## [10,] -1.332712093  1.28703233 -1.40675480  1.13970807 -1.14617432  0.834657877
## [11,]  0.386506559 -1.01877858  0.23563238 -0.70698224  0.26593586 -0.525494855
## [12,]  0.913312811 -0.45580577  0.44985948 -0.58029419  0.31589573 -0.500764724
## [13,] -0.962566814  0.25481874 -0.46286065 -0.12134198 -0.07079249 -0.282114247
## [14,] -1.116239701  0.64766959 -0.79042010  0.27561139 -0.37629668  0.002833529
## [15,] -1.560944777  2.07002906 -2.28316860  2.56590432 -2.62055586  2.682398165
## [16,] -0.053225363 -1.03268080  0.22486191 -0.70746087  0.26572564 -0.525506567
## [17,]  0.863086017 -0.53516378  0.40900453 -0.61083316  0.30141253 -0.509080851
## [18,] -1.196765150  0.87376012 -0.99724310  0.55029950 -0.60774291  0.238922615
## [19,] -0.334501033 -0.82397917  0.17164350 -0.68684274  0.26046639 -0.523833703
## [20,]  1.241019073  0.19485270  0.87503473 -0.17296239  0.56578990 -0.313515431
## [21,]  0.599912780 -0.86248133  0.27413140 -0.69281165  0.26932992 -0.524488959
## [22,]  1.630711299  1.26853987  1.84595839  1.11092200  1.65032659  0.802998379
## [23,]  0.113685482 -1.07626212  0.22914944 -0.70813357  0.26579673 -0.525516031
## [24,] -0.275138970 -0.88215889  0.19024189 -0.69548396  0.26301623 -0.524745564
## [25,]  1.291017724  0.31438686  0.96813025 -0.06771143  0.64126933 -0.247994065
## [26,]  1.463918615  0.76909984  1.35827647  0.41894361  1.02688862  0.122413814
## [27,]  0.217953768 -1.07314934  0.22932139 -0.70812715  0.26579699 -0.525516021
## [28,]  1.142743931 -0.02446388  0.71550100 -0.34151812  0.45276130 -0.405313221
## [29,] -0.639832403 -0.40525489 -0.01894586 -0.55867113  0.20489038 -0.494226527
## [30,] -0.149533681 -0.98033495  0.21548344 -0.70499377  0.26530948 -0.525420726
## [31,] -1.719282980  2.67891266 -3.04629323  3.95498098 -4.22516982  4.927179752
## [32,] -1.042531248  0.45291662 -0.62276733  0.06607736 -0.21016201 -0.156634713
## [33,]  0.470020323 -0.96925272  0.24527131 -0.70423920  0.26643450 -0.525384391
## [34,] -0.408257573 -0.74115716  0.14130183 -0.67068295  0.25499907 -0.521591392
## [35,] -0.335703181 -0.82272285  0.17121587 -0.68663147  0.26040020 -0.523808602
## [36,]  1.306446468  0.35235618  0.99856516 -0.03230312  0.66739560 -0.224662964
## [37,] -1.420561350  1.57518422 -1.71468663  1.61745152 -1.61637871  1.394901261
## [38,]  1.388711128  0.56342808  1.17502626  0.18190025  0.83236633 -0.070834423
## [39,] -1.209027127  0.90940870 -1.03099958  0.59669359 -0.64818221  0.281583169
## [40,]  0.002320563 -1.05382120  0.22750395 -0.70794394  0.26578208 -0.525514609
## [41,]  1.039447497 -0.23264591  0.57870640 -0.47211070  0.37361421 -0.463424783
## [42,] -1.279658435  1.12103318 -1.23756488  0.88939780 -0.91128505  0.567864796
## [43,]  1.000661480 -0.30490188  0.53485589 -0.51074167  0.35202677 -0.478026956
## [44,] -0.888695010  0.08400733 -0.33416755 -0.26212528  0.02691978 -0.364224112
```

```
## [45,]  1.065475772 -0.18234678  0.61036824 -0.44318197  0.39037851 -0.451666312
## [46,]  0.042732681 -1.06504106  0.22852680 -0.70808035  0.26579371 -0.525515817
## [47,]  0.091086961 -1.07386064  0.22906488 -0.70812918  0.26579658 -0.525516025
## [48,] -1.210149442  0.91268766 -1.03411982  0.60100310 -0.65195706  0.285584927
## [49,] -1.521016161  1.92497903 -2.11153521  2.27112743 -2.29945776  2.259066114
## [50,] -0.430509454 -0.71387809  0.13040808 -0.66436803  0.25267716 -0.520558019
## [51,]  0.271931310 -1.06237317  0.23004177 -0.70805545  0.26580162 -0.525515652
## [52,]  0.388914201 -1.01756030  0.23583317 -0.70693421  0.26594315 -0.525493517
## [53,]  1.700035061  1.49368368  2.08415964  1.47676230  2.00673454  1.223297024
## [54,]  0.640683063 -0.82150275  0.28755517 -0.68642530  0.27125805 -0.523783986
##                7           8
##  [1,]  0.26851984 -0.41236240
##  [2,]  0.21274528 -0.37970689
##  [3,]  0.52645633 -0.22700029
##  [4,]  0.59866968 -0.16630672
##  [5,] -1.60327760  1.38275260
##  [6,]  0.26933112 -0.41262234
##  [7,]  0.26996849 -0.41243470
##  [8,]  0.07752584 -0.27981412
##  [9,]  0.26934557 -0.41262492
## [10,] -0.80367167  0.53746123
## [11,]  0.26934821 -0.41262455
## [12,]  0.27935965 -0.40807747
## [13,]  0.12520345 -0.31681685
## [14,] -0.08668566 -0.14334816
## [15,] -2.65036445  2.57001433
## [16,]  0.26934454 -0.41262480
## [17,]  0.27555641 -0.40999059
## [18,] -0.27848499  0.02802522
## [19,]  0.26891075 -0.41249878
## [20,]  0.39203481 -0.33293198
## [21,]  0.26959008 -0.41255959
## [22,]  1.31328148  0.50809053
## [23,]  0.26934557 -0.41262492
## [24,]  0.26917074 -0.41258039
## [25,]  0.43732655 -0.29850357
## [26,]  0.72105849 -0.05916429
## [27,]  0.26934557 -0.41262492
## [28,]  0.33263244 -0.37522622
## [29,]  0.25618186 -0.40640905
## [30,]  0.26933031 -0.41262218
## [31,] -5.15224020  5.63779376
## [32,]  0.03522066 -0.24584160
## [33,]  0.26936783 -0.41262070
## [34,]  0.26817739 -0.41223465
## [35,]  0.26890318 -0.41249627
## [36,]  0.45391512 -0.28553542
## [37,] -1.33528377  1.09224125
## [38,]  0.56816246 -0.19220699
## [39,] -0.31431621  0.06111461
## [40,]  0.26934546 -0.41262491
## [41,]  0.29862856 -0.39712447
## [42,] -0.56237967  0.29749606
## [43,]  0.29076326 -0.40178327
```

```
## [44,]  0.18016001 -0.35727455
## [45,]  0.30519538 -0.39309193
## [46,]  0.26934556 -0.41262492
## [47,]  0.26934557 -0.41262492
## [48,] -0.31769383  0.06424905
## [49,] -2.20598512  2.05711061
## [50,]  0.26781117 -0.41209194
## [51,]  0.26934560 -0.41262492
## [52,]  0.26934841 -0.41262452
## [53,]  1.70796904  0.91567125
## [54,]  0.26979542 -0.41249379
## attr(,"degree")
## [1] 1 2 3 4 5 6 7 8
## attr(,"class")
## [1] "poly"   "matrix"
## attr(,"scaled:center")
##             1             2             3             4             5
## -4.541386e+00  8.978267e+02 -1.224838e+04  1.487124e+06 -3.242234e+07
##             6             7             8
##  2.953494e+09 -8.516708e+10  6.540112e+12
## attr(,"scaled:scale")
##            1            2            3            4            5            6
## 2.989571e+01 8.329950e+02 5.344499e+04 2.100059e+06 1.219817e+08 5.620178e+09
##            7            8
## 3.162000e+11 1.585002e+13
```

Pregunta 4

```
modeloP = lm(Yt ~ Xt + I(Xt^2) + I(Xt^3)+ I(Xt^4)+ I(Xt^5)+ I(Xt^6)+ I(Xt^7)+ I(Xt^8))
x_grid = seq(min(Xall), max(Xall), 0.1)
y_grid = modeloP$coefficients[[1]] + modeloP$coefficients[[2]]*x_grid^1 + modeloP$coefficients[[3]]*x_g
plot(Xall, Yall)
lines(x_grid, y_grid, col = "dark green")
```



Realmente no aprecio problemas de varianza ni de sesgo.

Pregunta 5

```
modelos2 = seq(1:12)
modelos2[1] = Yt[1]
modelos2[2] = lm(Yt[1:2] ~ Xt[1:2])
```

```
## Warning in modelos2[2] = lm(Yt[1:2] ~ Xt[1:2]): número de items para para
## sustituir no es un múltiplo de la longitud del reemplazo
```

```
modelos2[3] = lm(Yt[1:3] ~ Xt[1:3] + I(Xt[1:3]^2))
```

```
## Warning in modelos2[3] = lm(Yt[1:3] ~ Xt[1:3] + I(Xt[1:3]^2)): número de items
## para para sustituir no es un múltiplo de la longitud del reemplazo
```

```
modelos2[4] = lm(Yt[1:4] ~ Xt[1:4] + I(Xt[1:4]^2) + I(Xt[1:4]^3))
```

```
## Warning in modelos2[4] = lm(Yt[1:4] ~ Xt[1:4] + I(Xt[1:4]^2) + I(Xt[1:4]^3)):
## número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
modelos2[5] = lm(Yt[1:5] ~ Xt[1:5] + I(Xt[1:5]^2) + I(Xt[1:5]^3) + I(Xt[1:5]^4))
```

```
## Warning in modelos2[5] = lm(Yt[1:5] ~ Xt[1:5] + I(Xt[1:5]^2) + I(Xt[1:5]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
modelos2[6] = lm(Yt[1:6] ~ Xt[1:6] + I(Xt[1:6]^2) + I(Xt[1:6]^3) + I(Xt[1:6]^4) + I(Xt[1:6]^5))
```

```
## Warning in modelos2[6] = lm(Yt[1:6] ~ Xt[1:6] + I(Xt[1:6]^2) + I(Xt[1:6]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
modelos2[7] = lm(Yt[1:7] ~ Xt[1:7] + I(Xt[1:7]^2) + I(Xt[1:7]^3) + I(Xt[1:7]^4) + I(Xt[1:7]^5) + I(Xt[1
```

```
## Warning in modelos2[7] = lm(Yt[1:7] ~ Xt[1:7] + I(Xt[1:7]^2) + I(Xt[1:7]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
modelos2[8] = lm(Yt[1:8] ~ Xt[1:8] + I(Xt[1:8]^2) + I(Xt[1:8]^3) + I(Xt[1:8]^4) + I(Xt[1:8]^5) + I(Xt[1
```

```
## Warning in modelos2[8] = lm(Yt[1:8] ~ Xt[1:8] + I(Xt[1:8]^2) + I(Xt[1:8]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
modelos2[9] = lm(Yt[1:9] ~ Xt[1:9] + I(Xt[1:9]^2) + I(Xt[1:9]^3) + I(Xt[1:9]^4) + I(Xt[1:9]^5) + I(Xt[1
```

```
## Warning in modelos2[9] = lm(Yt[1:9] ~ Xt[1:9] + I(Xt[1:9]^2) + I(Xt[1:9]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
modelos2[10] = lm(Yt[1:10] ~ Xt[1:10] + I(Xt[1:10]^2) + I(Xt[1:10]^3) + I(Xt[1:10]^4) + I(Xt[1:10]^5) +
```

```
## Warning in modelos2[10] = lm(Yt[1:10] ~ Xt[1:10] + I(Xt[1:10]^2) + I(Xt[1:10]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
modelos2[11] = lm(Yt[1:11] ~ Xt[1:11] + I(Xt[1:11]^2) + I(Xt[1:11]^3) + I(Xt[1:11]^4) + I(Xt[1:11]^5) +
```

```
## Warning in modelos2[11] = lm(Yt[1:11] ~ Xt[1:11] + I(Xt[1:11]^2) + I(Xt[1:11]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

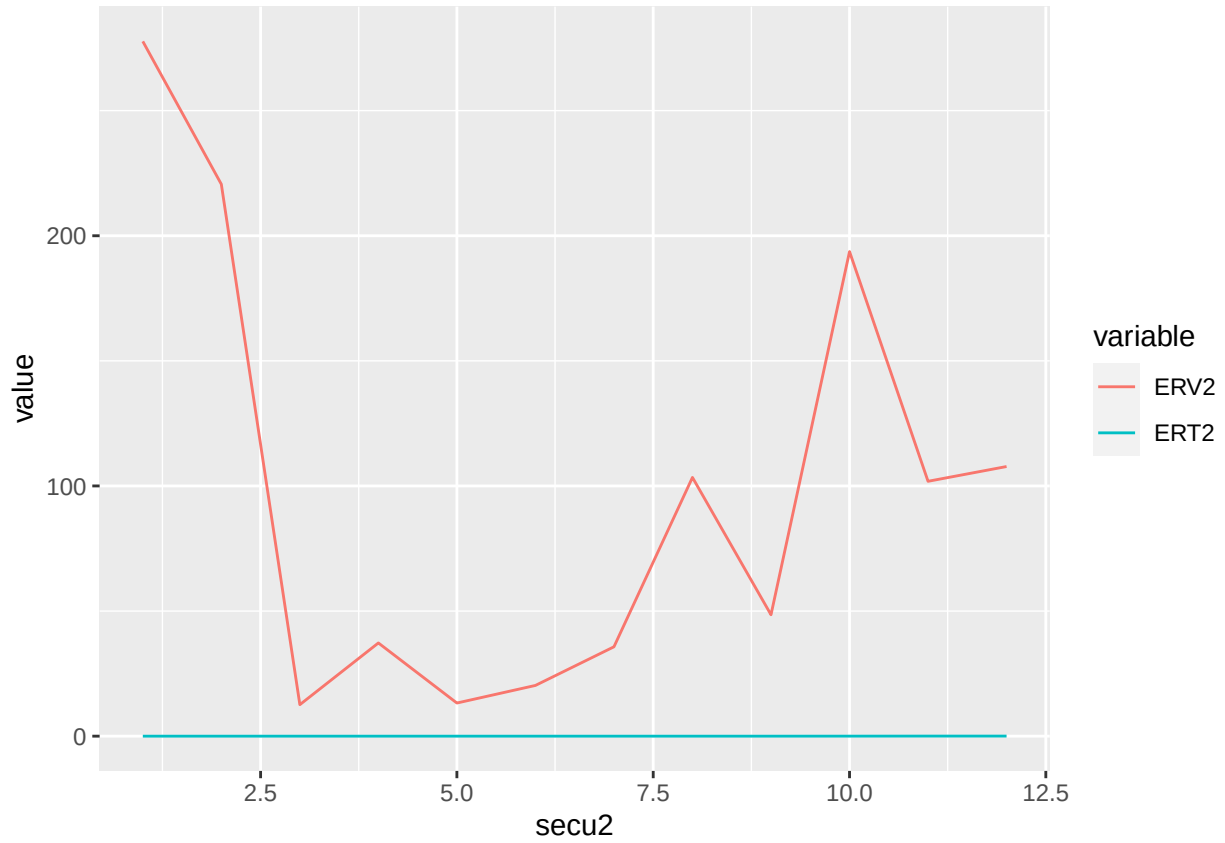
```
modelos2[12] = lm(Yt[1:12] ~ Xt[1:12] + I(Xt[1:12]^2) + I(Xt[1:12]^3) + I(Xt[1:12]^4) + I(Xt[1:12]^5) +
```

```
## Warning in modelos2[12] = lm(Yt[1:12] ~ Xt[1:12] + I(Xt[1:12]^2) + I(Xt[1:12]^3)
## + : número de items para para sustituir no es un múltiplo de la longitud del
## reemplazo
```

```
ERT2 = rep(0,12)
ERV2 = rep(0,12)

#ERT2


for (j in 1:2){
  ERT2[2] = ERT2[2] + (Yt[j]-(modelos2[[2]][[1]]+Xt[j]*modelos2[[2]][[2]]))^2
}
ERT2[2]= ERT2[2]/21


for (j in 1:3){
  ERT2[3] = ERT2[3] + (Yt[j]-(modelos2[[3]][[1]]+Xt[j]*modelos2[[3]][[2]]+Xt[j]^2*modelos2[[3]][[3]]))^
}
ERT2[3]= ERT2[3]/21

for (j in 1:4){
  ERT2[4] = ERT2[4] + (Yt[j]-(modelos2[[4]][[1]]+Xt[j]*modelos2[[4]][[2]]+Xt[j]^2*modelos2[[4]][[3]]+Xt
}
ERT2[4]= ERT2[4]/21

for (j in 1:5){
  ERT2[5] = ERT2[5] + (Yt[j]-(modelos2[[5]][[1]]+Xt[j]*modelos2[[5]][[2]]+Xt[j]^2*modelos2[[5]][[3]]+Xt
}
ERT2[5]= ERT2[5]/21

for (j in 1:6){
  ERT2[6] = ERT2[6] + (Yt[j]-(modelos2[[6]][[1]]+Xt[j]*modelos2[[6]][[2]]+Xt[j]^2*modelos2[[6]][[3]]+Xt
}
ERT2[6]= ERT2[6]/21


for (j in 1:7){
  ERT2[7] = ERT2[7] + (Yt[j]-(modelos2[[7]][[1]]+Xt[j]*modelos2[[7]][[2]]+Xt[j]^2*modelos2[[7]][[3]]+Xt
}
ERT2[7]= ERT2[7]/21

for (j in 1:8){
  ERT2[8] = ERT2[8] + (Yt[j]-(modelos2[[8]][[1]]+Xt[j]*modelos2[[8]][[2]]+Xt[j]^2*modelos2[[8]][[3]]+Xt
}
ERT2[8]= ERT2[8]/21

for (j in 1:9){
  ERT2[9] = ERT2[9] + (Yt[j]-(modelos2[[9]][[1]]+Xt[j]*modelos2[[9]][[2]]+Xt[j]^2*modelos2[[9]][[3]]+Xt
}
ERT2[9]= ERT2[9]/21

for (j in 1:10){
  ERT2[10] = ERT2[10] + (Yt[j]-(modelos2[[10]][[1]]+Xt[j]*modelos2[[10]][[2]]+Xt[j]^2*modelos2[[10]][[3
}
ERT2[10]= ERT2[10]/21
```

```
for (j in 1:11){
  ERT2[11] = ERT2[11] + (Yt[j]-(modelos2[[11]][[1]]+Xt[j]*modelos2[[11]][[2]]+Xt[j]^2*modelos2[[11]][[3]
}
ERT2[11]= ERT2[11]/21

for (j in 1:12){
  ERT2[12] = ERT2[12] + (Yt[j]-(modelos2[[12]][[1]]+Xt[j]*modelos2[[12]][[2]]+Xt[j]^2*modelos2[[12]][[3]
}
ERT2[12]= ERT2[12]/21


#ERV2

for (j in 1:21){
  ERV2[1] = ERV2[1] + (Yv[j]-(modelos2[[1]][[1]]))^2
}
ERV2[1]= ERV2[1]/21


for (j in 1:21){
  ERV2[2] = ERV2[2] + (Yv[j]-(modelos2[[2]][[1]]+Xv[j]*modelos2[[2]][[2]]))^2
}
ERV2[2]= ERV2[2]/21


for (j in 1:21){
  ERV2[3] = ERV2[3] + (Yv[j]-(modelos2[[3]][[1]]+Xv[j]*modelos2[[3]][[2]]+Xv[j]^2*modelos2[[3]][[3]]))^
}
ERV2[3]= ERV2[3]/21

for (j in 1:21){
  ERV2[4] = ERV2[4] + (Yv[j]-(modelos2[[4]][[1]]+Xv[j]*modelos2[[4]][[2]]+Xv[j]^2*modelos2[[4]][[3]]+Xv
}
ERV2[4]= ERV2[4]/21

for (j in 1:21){
  ERV2[5] = ERV2[5] + (Yv[j]-(modelos2[[5]][[1]]+Xv[j]*modelos2[[5]][[2]]+Xv[j]^2*modelos2[[5]][[3]]+Xv
}
ERV2[5]= ERV2[5]/21

for (j in 1:21){
  ERV2[6] = ERV2[6] + (Yv[j]-(modelos2[[6]][[1]]+Xv[j]*modelos2[[6]][[2]]+Xv[j]^2*modelos2[[6]][[3]]+Xv
}
ERV2[6]= ERV2[6]/21


for (j in 1:21){
  ERV2[7] = ERV2[7] + (Yv[j]-(modelos2[[7]][[1]]+Xv[j]*modelos2[[7]][[2]]+Xv[j]^2*modelos2[[7]][[3]]+Xv
}
ERV2[7]= ERV2[7]/21

for (j in 1:21){
  ERV2[8] = ERV2[8] + (Yv[j]-(modelos2[[8]][[1]]+Xv[j]*modelos2[[8]][[2]]+Xv[j]^2*modelos2[[8]][[3]]+Xv
```

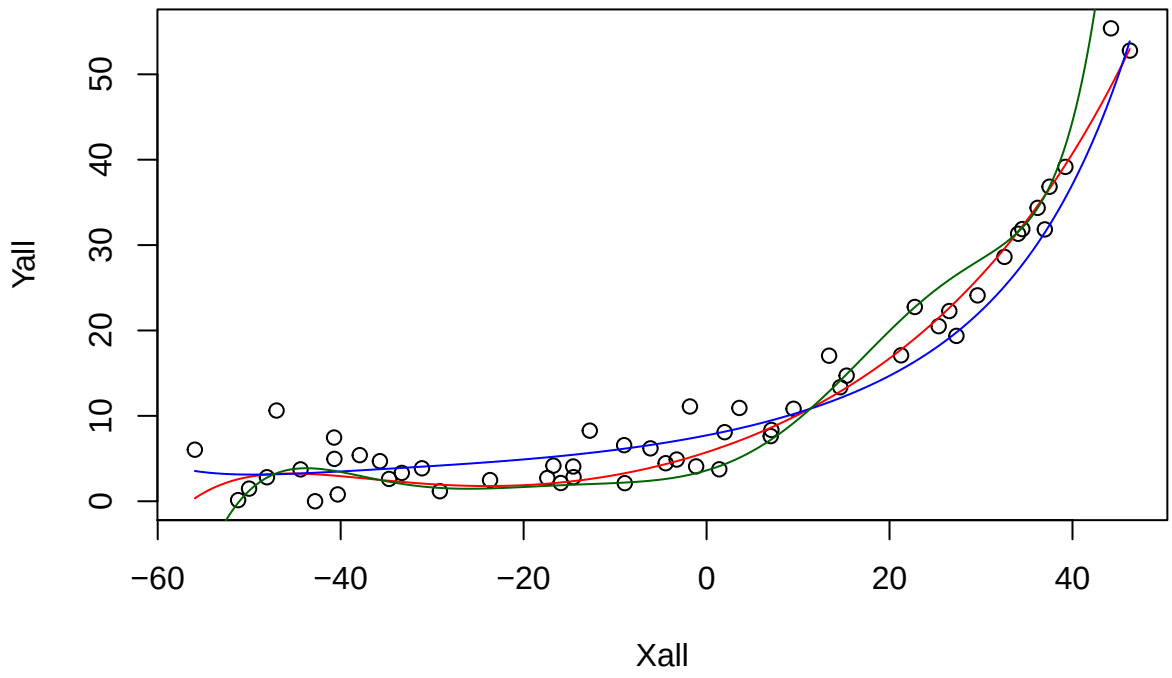
```
}
ERV2[8]= ERV2[8]/21

for (j in 1:21){
  ERV2[9] = ERV2[9] + (Yv[j]-(modelos2[[9]][[1]]+Xv[j]*modelos2[[9]][[2]]+Xv[j]^2*modelos2[[9]][[3]]+Xv
}
ERV2[9]= ERV2[9]/21

for (j in 1:21){
  ERV2[10] = ERV2[10] + (Yv[j]-(modelos2[[10]][[1]]+Xv[j]*modelos2[[10]][[2]]+Xv[j]^2*modelos2[[10]][[3
}
ERV2[10]= ERV2[10]/21

for (j in 1:21){
  ERV2[11] = ERV2[11] + (Yv[j]-(modelos2[[11]][[1]]+Xv[j]*modelos2[[11]][[2]]+Xv[j]^2*modelos2[[11]][[3
}
ERV2[11]= ERV2[11]/21

for (j in 1:21){
  ERV2[12] = ERV2[12] + (Yv[j]-(modelos2[[12]][[1]]+Xv[j]*modelos2[[12]][[2]]+Xv[j]^2*modelos2[[12]][[3
}
ERV2[12]= ERV2[12]/21

secu2 = seq(1:12)

df3 = data.frame(secu2, ERV2, ERT2)
df4 = melt(data = df3, id.vars = "secu2")
ggplot(data = df4, aes(x = secu2, y = value, colour = variable)) + geom_line()
```



Noto problemas de varianza, ya que los errores de la base de validación y la de entrenamiento no están corvengiendo al mismo valor. Los errores de la base de validación son notoriamente más altos.

Pregunta 6

```
mmo= model.matrix(Yt ~ Xt + I(Xt^2) + I(Xt^3)+ I(Xt^4)+ I(Xt^5)+ I(Xt^6)+ I(Xt^7)+ I(Xt^8))[,-1]
modeloR = glmnet(mmo,Yt,lambda = 10/(2*length(Xt)),alpha = 0)
modeloR2 = glmnet(mmo,Yt,lambda = 100/(2*length(Xt)),alpha = 0)
coR = coef(modeloR)
coR2 = coef(modeloR2)
yr_grid = coR[1] + coR[2]*x_grid^1 + coR[3]*x_grid^2 + coR[4]*x_grid^3 + coR[5]*x_grid^4 + coR[6]*x_gri
yr2_grid = coR2[1] + coR2[2]*x_grid^1 + coR2[3]*x_grid^2 + coR2[4]*x_grid^3 + coR2[5]*x_grid^4 + coR2[6
plot(Xall, Yall)
lines(x_grid, yr_grid, col = "red")
lines(x_grid, yr2_grid, col = "blue")
lines(x_grid, y_grid, col = "dark green")
```



Podemos ver que la curva penalizada por lambda = 100 (azul), a simple vista, se aleja demasiado de los datos. Por otro lado, la curva con lambda = 10 (roja) sí se ajusta bien a los datos y se puede apreciar un menor overfitting que en la curva polinómica sin penalizar (verde oscuro), por lo que, teniendo que elegir uno de los dos lambda, me quedaría con lambda = 10.

Pregunta 7

```
Ete1 = 0

for (j in 1:21){
  Ete1 = Ete1 + (Yte[j]-(modelo$coefficients[[1]]+Xte[j]*modelo$coefficients[[2]]))^2
}
Ete1= Ete1/21

Ete2 = 0

for (j in 1:21){
  Ete2 = Ete2 + (Yte[j]-(modeloP$coefficients[[1]]+Xte[j]*modeloP$coefficients[[2]]+Xte[j]^2*modeloP$co
}
Ete2= Ete2/21

Ete31 = 0

for (j in 1:21){
```

```
  Ete31 = Ete31 + (Yte[j]-(coR[1]+Xte[j]*coR[2]+Xte[j]^2*coR[3]+Xte[j]^3*coR[4]+Xte[j]^4*coR[5]+Xte[j]^
}
Ete31= Ete31/21

Ete32 = 0

for (j in 1:21){
  Ete32 = Ete32 + (Yte[j]-(coR2[1]+Xte[j]*coR2[2]+Xte[j]^2*coR2[3]+Xte[j]^3*coR2[4]+Xte[j]^4*coR2[5]+Xt
}
Ete32= Ete32/21
```

El error del modelo lineal con una variable de entrada es 65.0114985; el error del modelo polinomial sin penalizar es 34.24008; el error del modelo penalizado con lambda = 10 es 8.8422148 y el con lambda = 100 es 9.5283383. Podemos apreciar que el modelo con menor error es el penalizado con lambda = 10, que es justo el que había seleccionado en la pregunta anterior. Con todo esto, el mejor modelo de los que cree es el penalizado con lambda = 10, que era lo esperable.

Parte 2

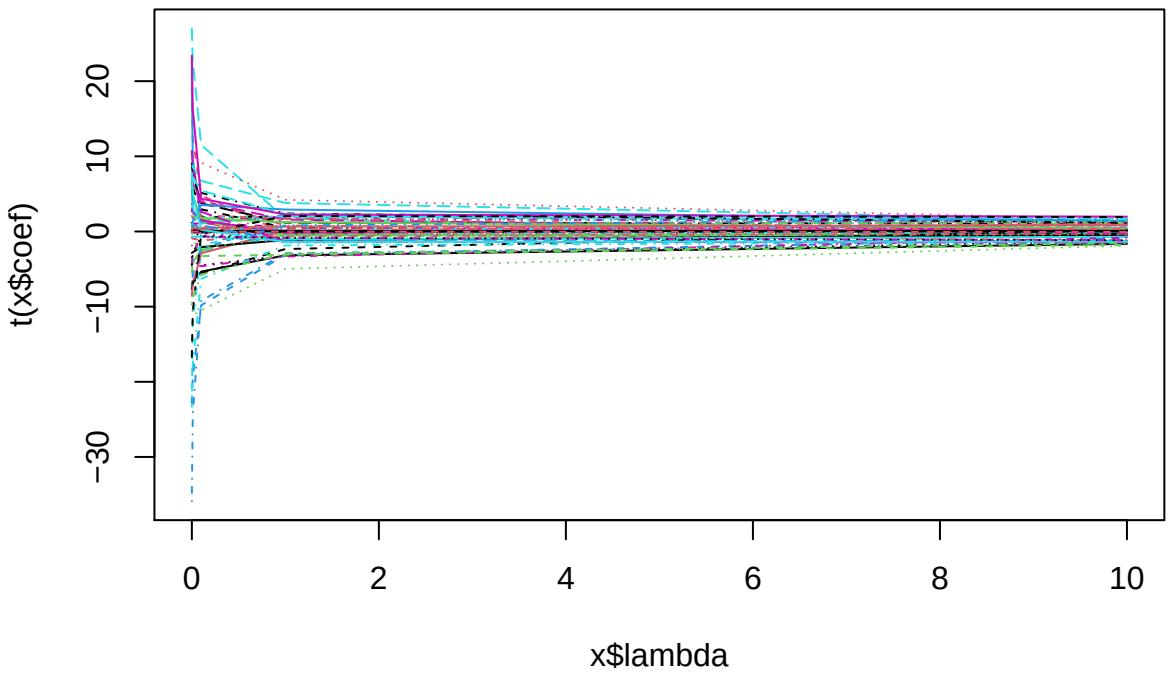
```
datos2 = read.csv("FINAL_USO.csv")
datos2$Date = NULL
datos2$Open = NULL
datos2$High = NULL
datos2$Low = NULL
datos2$Close = NULL
datos2$Volume = NULL

set.seed(19081998)
```

Pregunta 1

```
sepa = c(Entrenamiento = .6, Validacion = .2, Testeo = .2)

sepaR = sample(cut(seq(nrow(datos2)), nrow(datos2)*cumsum(c(0,sepa)),labels = names(sepa)))

datos2F = split(datos2, sepaR)
```

Pregunta 2

```
lambdas = c(0.00001,0.0001,0.001,0.01,0.1,1,10)
modeloR = lm.ridge(Adj.Close~ ., data=datos2F$Entrenamiento, lambda = lambdas)
plot(modeloR)
```

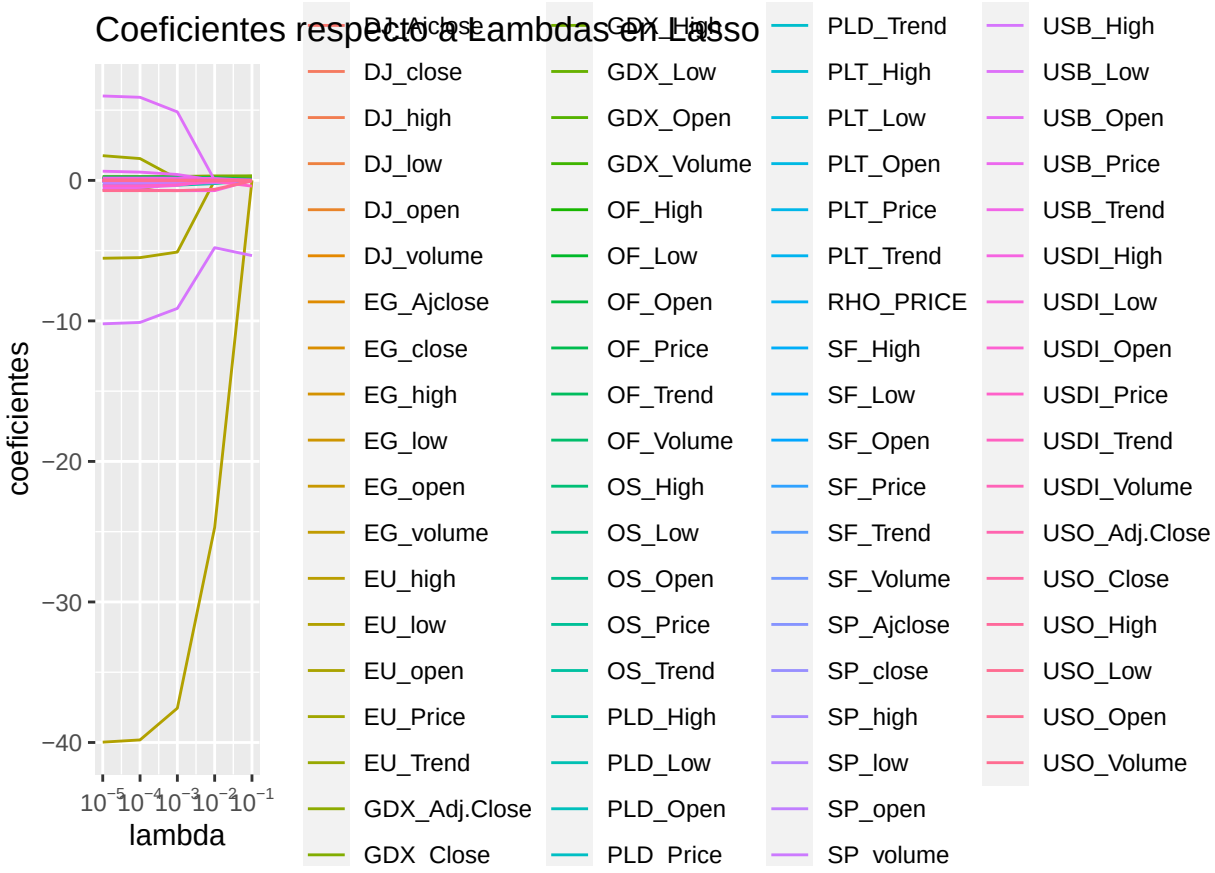


Entre más grande el Lambda, más pequeños los coeficientes, o sea, pierden importancia. Debería tomar un lambda "pequeño".

Pregunta 3

```
lambdas2 = c(0.00001,0.0001,0.001,0.01,0.1)

mmo= model.matrix(Adj.Close ~ . , data=datos2F$Entrenamiento)[,-1]
modeloL = glmnet(mmo,datos2F$Entrenamiento$Adj.Close,lambda = lambdas2,alpha = 1)
coL = coef(modeloL)

regularizacion <- modeloL$beta %>%
  as.matrix() %>%
  t() %>%
  as_tibble() %>%
  mutate(lambda = modeloL$lambda)

regularizacion <- regularizacion %>%
  pivot_longer(
    cols = !lambda,
    names_to = "predictor",
    values_to = "coeficientes"
  )
```

```
regularizacion %>%
  ggplot(aes(x = lambda, y = coeficientes, color = predictor)) +
  geom_line() +
  scale_x_log10(
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))
  ) +
  labs(title = "Coeficientes respecto a Lambdas en Lasso")
```



No todos los coeficientes se comportan de la misma forma respecto al lambda. Nuevamente, debería elegir un lambda pequeño.

Pregunta 4

```
foldR = cv.glmnet(x = mmo, y = datos2F$Entrenamiento$Adj.Close, alpha  = 0, nfolds = 10,type.measure = 
foldR$lambda.min
```

```
## [1] 1.703722
```

No alcancé a hacer la validación cruzada con la base de validación, pero con el 10-fold se llega a un lambda óptimo de 1.7037223.

Pregunta 5

```
foldL = cv.glmnet(x = mmo, y = datos2F$Entrenamiento$Adj.Close, alpha  = 1, nfolds = 10,type.measure =
foldL$lambda.min
```

```
## [1] 0.004319557
```

Idem que la anterior. El lambda óptimo es 0.0043196.

Pregunta 6

```
modeloRO = lm.ridge(Adj.Close~ ., data=datos2F$Entrenamiento, lambda = foldR$lambda.min)
modeloRO2 = glmnet(mmo,datos2F$Entrenamiento$Adj.Close,lambda = foldL$lambda.min,alpha = 0)
modeloLO = glmnet(mmo,datos2F$Entrenamiento$Adj.Close,lambda = foldL$lambda.min,alpha = 1)
mmoT= model.matrix(Adj.Close ~ . , data=datos2F$Testeo)[,-1]

prediccionesR = predict(modeloRO2, newx = mmoT)
prediccionesL = predict(modeloLO, newx = mmoT)
ETRO= 0
for (j in 1:length(prediccionesR)){
  ETRO = ETRO + (datos2F$Testeo$Adj.Close[j]-prediccionesR[j])^2
}
ETRO = ETRO/344


ETLO= 0
for (j in 1:length(prediccionesL)){
  ETLO = ETLO + (datos2F$Testeo$Adj.Close[j]-prediccionesL[j])^2
}
ETLO = ETLO/344


rango = max(datos2$Adj.Close)-min(datos2$Adj.Close)
```

El modelo Ridge con lambda óptimo tiene un error cuadrático medio (4.5485852) levemente menor al del modelo Lasso con lambda óptimo (4.6209478). En general, son muy buenos errores, tomando en cuenta que la variable respuesta tiene un rango de 73.110001.