02_LASSO_example

June 25, 2019

1 Using the LASSO for inference and prediction

- Stepwise linear regression is time consuming - the p-values are unstable and are path-dependent
- Would be nice to have the optimal subset of variables selected in an unbiased way
- Use LASSO/Ridge/ElasticNet for wide data
- Can use LASSO for unbiased feature selection
- LASSO uses regularization which penalizes high coefficients
- Lambda parameter analogy to garden hose - most important features start to trickle through as you open the valve

```
In [1]: # Package "glmnet" contains the LASSO function
        # install.packages( 'glmnet' )
        library( glmnet )

        # Package "mice" is for imputation of missing data
        #install.packages( 'mice' )
        library( mice )

        # Metapackage "tidyverse" imports libraries
        # for data manipulation (dplyr) and plotting (ggplot2)
        library( tidyverse )

        # Package "readxl" has the read_excel() function
        library( readxl )

        # Package "skimr" has excellent descriptive statistics
        # function skim_to_wide()
        library( skimr )

        # Package "GGally" has ggplot2-style scatterplot matrices
        library( GGally )

        # Package "ggfortify" has ggplot2-style regression diagnostic plots
        #install.packages( 'ggfortify' )
        library( ggfortify )
```

```r
# Package tictoc has functions to time function calls
library( tictoc )

# Package "glmnetUtils" allows the use of R formulas for
# specifying glmnet models (as opposed to converting to matrices)
#library(devtools)
#install_github("hong-revo/glmnetUtils")
library(glmnetUtils)

# Package "broom" for tidy() function for pulling
# regression coefficients from glmnet models
library( broom )

# Package "rsample" for train/test split utilities
library( rsample )
```

```
Loading required package: Matrix
Loading required package: foreach
Loaded glmnet 2.0-18

Loading required package: lattice

Attaching package: mice

The following objects are masked from package:base:

    cbind, rbind

Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Registered S3 method overwritten by 'rvest':
  method            from
  read_xml.response xml2
 Attaching packages  tidyverse 1.2.1
 ggplot2 3.1.1       purrr   0.3.2
 tibble  2.1.1       dplyr   0.8.1
 tidyr   0.8.3       stringr 1.4.0
 readr   1.3.1       forcats 0.4.0
 Conflicts  tidyverse_conflicts()
 purrr::accumulate() masks foreach::accumulate()
 tidyr::complete()   masks mice::complete()
 tidyr::expand()     masks Matrix::expand()
 dplyr::filter()     masks stats::filter()
 dplyr::lag()        masks stats::lag()
 purrr::when()       masks foreach::when()
```

```
Attaching package: skimr

The following object is masked from package:stats:

    filter

Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2

Attaching package: GGally

The following object is masked from package:dplyr:

    nasa


Attaching package: glmnetUtils

The following objects are masked from package:glmnet:

    cv.glmnet, glmnet
```

2 Load data

- NYtowns dataset - marketing data
- [Product] Penetration is target variable
- Many other variables, mostly demographic data

```
In [2]: data <- read_excel( 'NYTowns.xlsx' )
```

```
In [3]: data %>% dim
```

1. 1006 2. 248

```
In [4]: data <- data %>%
            #select( -starts_with( 'Anc') ) %>%
            select( -c( 'GEO_ID', "GEO_NAME", "NAME") )
```

3 Exploratory: univariate

- Distribution of Penetration is right skew and non-negative; not a count, but more like a rate

```
In [5]: # A Jupyter Notebook-specific directive
        # sets the maximum number of rows to something above what is needed
```

```
        # to show everything
        options( repr.matrix.max.rows=300 )
```

```
In [6]: skim_to_wide( data ) %>% select( -type, -n )
```

variable <chr>	missing <chr>	complete <chr>	mean <chr>	sd <chr>	p0 <chr>	p25 <chr
AncArab	0	1006	0.28	0.43	0	0
AncCzech	0	1006	0.49	0.5	0	0.1
AncDanish	0	1006	0.37	0.61	0	0.1
AncDutch	0	1006	4.16	2.87	0	2.3
AncEnglish	0	1006	14.37	5.39	0	10.6
AncFrCanad	0	1006	2.07	2.35	0	0.8
AncFrench	0	1006	6.58	6.47	0	3
AncGerman	0	1006	19.89	8.34	0	14.8
AncGreek	0	1006	0.34	0.49	0	0
AncHungary	0	1006	0.61	0.67	0	0.2
AncIrish	0	1006	17.73	5.12	0	14.3
AncItalian	0	1006	11.23	7.11	0	6.12
AncLithu	0	1006	0.26	0.34	0	0
AncNorweg	0	1006	0.67	0.62	0	0.2
AncOthr	0	1006	11.62	8.24	1.6	7.6
AncPolish	0	1006	5.91	4.57	0	3.1
AncPortug	0	1006	0.15	0.25	0	0
AncRussian	0	1006	0.93	1.26	0	0.2
AncScot	0	1006	2.37	1.32	0	1.5
AncScotIre	0	1006	1.39	0.8	0	0.9
AncSlovak	0	1006	0.27	0.57	0	0
AncSubSah	0	1006	0.12	0.26	0	0
AncSwedish	0	1006	1.35	2.35	0	0.5
AncSwiss	0	1006	0.43	0.64	0	0.1
AncUkraine	0	1006	0.68	0.83	0	0.2
AncUS	0	1006	6.85	3.66	0	4.1
AncWelsh	0	1006	1.22	1.42	0	0.5
AncWIndian	0	1006	0.25	0.91	0	0
AreaLand	0	1006	1.2e+08	9.6e+07	230812	8.2e
AreaWater	0	1006	9e+06	5e+07	0	1587
BadKitchen	0	1006	0.69	1.35	0	0.1
BadPlumbing	0	1006	0.75	1.35	0	0.2
BoatRVVan	0	1006	0.45	1.14	0	0
BornAfrica	14	992	1.26	3.36	0	0
BornAsia	14	992	17.28	17.1	0	0
BornEurope	14	992	49.57	23.92	0	32.6
BornLatAmer	14	992	14.48	18.74	0	0
BornNorAmer	14	992	16.79	20.57	0	2.1
BornOceania	14	992	0.62	2.82	0	0
BuiltAfter40	0	1006	67.87	13.23	28.8	59.3
BuiltAfter60	0	1006	51.62	13.99	10.7	43.4
BuiltAfter70	0	1006	40.81	13.1	5.3	33.0
BuiltAfter80	0	1006	26.13	9.98	1.3	19.7
BuiltAfter90	0	1006	13.22	5.93	0.2	9.2
BuiltAfter95	0	1006	6.43	3.62	0	4.1
BuiltAfter99	0	1006	1.48	1.21	0	0.6
CitizenNative	0	1006	96.41	4.61	53.9	95.9
CitizenNatr	0	1006	1.97	2.12	0	0.7
CitizenNot	0	1006	1.62	2.75	0	0.3
CommuteAtHome	0	1006	4.45	2.96	0	2.6
CommuteAvgTravTime	0	1006	25.91	6.03	6	21.6

4 Optional: impute missing data

```
In [7]: data %>% is.na %>% sum
```

190

```
In [8]: tic()
        imputation_model <- mice( data, method='cart', m=1, maxit=1 )
        toc()
```

```
 iter imp variable
  1   1  GPGuardP  BornEurope  BornAsia  BornAfrica  BornOceania  BornLatAmer  BornNorAmer  In

Warning message:
Number of logged events: 23

27.761 sec elapsed
```

```
In [9]: #glimpse( imputation_model )

In [10]: imputed_data <- mice::complete( imputation_model )

In [11]: imputed_data %>% is.na %>% sum
```

0

5 To evaluate predictions, split into train and test set

```
In [12]: set.seed( 42 )
         data_splitter <- initial_split( imputed_data, prop=0.8 )
         train_data <- training( data_splitter )
         test_data <- testing( data_splitter )
```

6 Perform LASSO

- Plain vanilla model (gaussian family w/identity link function)
  - Baseline: fitting linear model to poisson-style data
- Vary lambda regularization parameter, observe coefficient paths as lambda varies
  - By default, it will generate 100 different models with 100 different lambdas
  - A reasonable range of lambda vales are selected for you by default, also you can specify your own (or a range of lambdas)
- LASSO corresponds with glmnet with argument $\alpha = 1$ which is default
- glmnet standardized variables by default, so regression coefficients β_i is interpretable as relative importances

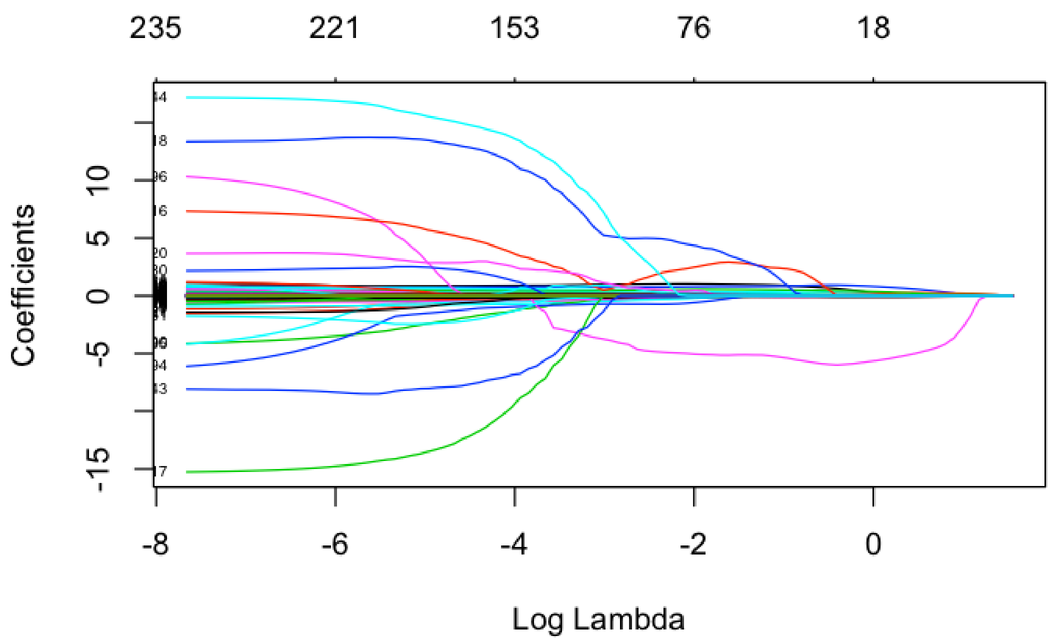
- Use `dfmax` argument to limit number of variables included in the model

```
In [13]: glmnet_lm_result <- glmnet( Penetration ~ ., data=train_data )
```

```
In [14]: print( glmnet_lm_result )
```

```
Call:
glmnet.formula(formula = Penetration ~ ., data = train_data)

Model fitting options:
    Sparse model matrix: FALSE
    Use model.frame: FALSE
    Alpha: 1
    Lambda summary:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000470 0.004705 0.047063 0.529144 0.470507 4.701200
```

```
In [15]: options(repr.plot.width=6, repr.plot.height=4)
         plot( glmnet_lm_result, xvar='lambda', label=TRUE )
         # can also use ggfortify version, which doesn't work very well with many variables
         #options(repr.plot.width=30, repr.plot.height=30 )
         #autoplot( glmnet_lm_result, label=FALSE )
         #plot_glmnet( glmnet_lm_result, xvar='lambda', label=TRUE )
```



6.1 How many times does a given variable appear in a model?

```
In [16]: glmnet_lm_result %>%                    # Take the glmnet result object
             coef %>%                            # get all the coefficients for all 100 models (l
             t %>%                               # transpose so the variables are in the columns
             as.matrix %>% as.data.frame %>%     # convert to something the tidyverse can manipul
             map_int( ~ sum( . != 0 ) ) %>%      # For each column, return a count
             sort( decreasing = TRUE ) %>%       # sort from highest to lowest
             head( 40 )  %>%                     # show the top 20 used variables
             print
```

```
      (Intercept)   HouseMultiFamily          AncItalian          ValueLT50K
              100                 98                  97                  95
      BadPlumbing          NoCashRent        LessEq2Rooms            EduHSDip
               94                 94                  92                  91
   JobAgriculture MarFemaleDivorcees        WorkClassSelf        BuiltAfter90
               89                 88                  87                  86
         AncSwiss       CommuteAtHome            PopRural         JobConstruct
               85                 82                  82                  81
     BuiltAfter70             SchNurs      JobOfficeSales         EduColNoDeg
               80                 79                  79                  78
       BadKitchen CommuteAvgTravTime       CostDivIncLT15             VehicGT1
               78                 78                  78                  78
  IndConstruction            Resid5Yrs           IndAgric       PovIndSeniors
               77                 76                  75                  75
     IncAvgSocSec       IncRetirement      CostDivIncLT25            MarWidow
               74                 74                  74                  72
    NativityOutUS         MortageLT500          JobService           IncSupSec
               72                 72                  71                  71
         AreaLand             SchKind             AncOthr      IndPublicAdmin
               71                 70                  70                  70
```

7 Which lambda gives the best model?

- BEST lambda is given by "lambda.min": the at which the minimal MSE is achieved
- Use `cv.glmnet()` - Perform CROSS-VALIDATION LASSO
- Again, plain vanilla model (gaussian family w/identity link function)
- 10-fold cross validation to find optimal lambda parameter

```
In [17]: tic()
         glmnet_cv_result <- cv.glmnet( Penetration ~ ., data=train_data )
         toc()
```

```
1.736 sec elapsed
```

```
In [18]: print( glmnet_cv_result )
```

```
Call:
cv.glmnet.formula(formula = Penetration ~ ., data = train_data)

Model fitting options:
    Sparse model matrix: FALSE
    Use model.frame: FALSE
    Number of crossvalidation folds: 10
    Alpha: 1
    Deviance-minimizing lambda: 0.2182104  (+1 SE): 0.6071833
```

In [19]:
```
glmnet_cv_result$lambda.min
```

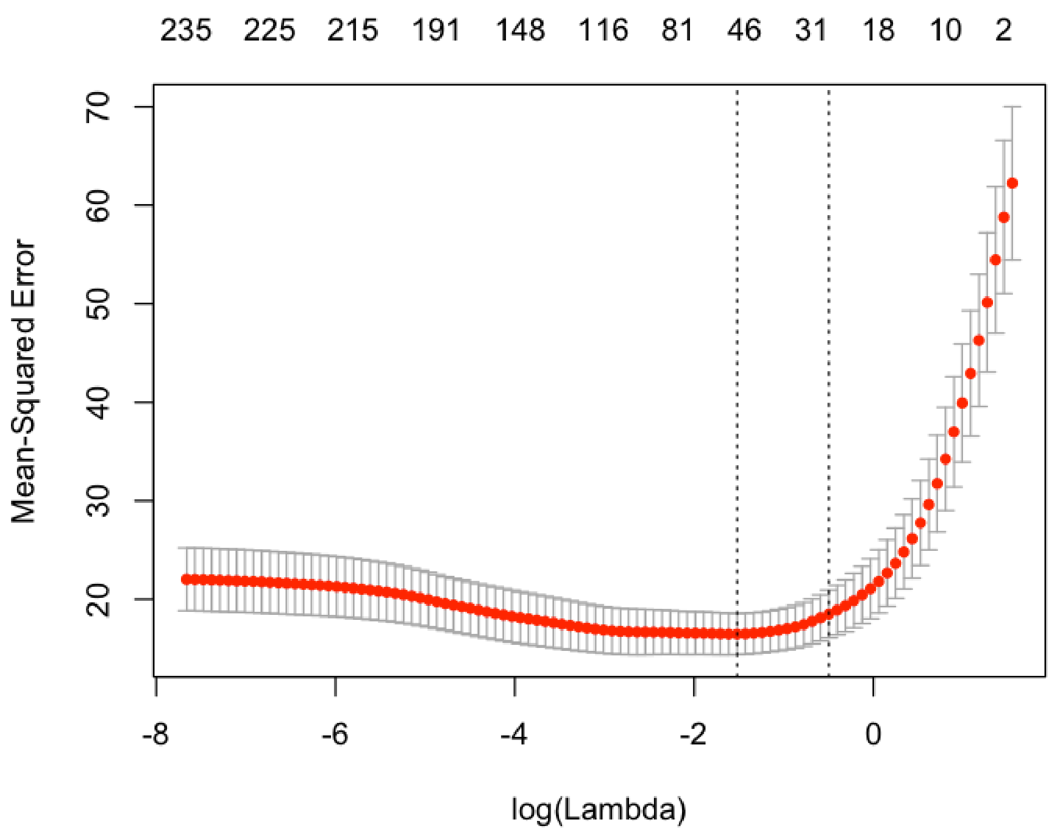
0.218210370672902

In [20]:
```
options(repr.plot.width=6, repr.plot.height=5)
```

In [21]:
```
log( glmnet_cv_result$lambda.min )
```

-1.52229567842249

In [22]:
```
plot( glmnet_cv_result )
```



7.1 Given the best lambda, what are the regression coefficients?

- Remember, glmnet regression coefficients are standardized
  - Betas aren't in original units
  - Betas double as feature importances
  - To keep original units, use argument standardize=False in glmnet function call

```
In [23]: lm_coefs <- glmnet_cv_result %>%
             coef( s = "lambda.min" ) %>%     # Get the betas from the best model
             tidy %>%                         # Put betas into a data.frame
             select( -column ) %>%            # prune unimportant column named "column"
             arrange( desc( abs( value ) ) ) # Sort by absolute value
```

```
Warning message:
'tidy.dgCMatrix' is deprecated.
See help("Deprecated")Warning message:
'tidy.dgTMatrix' is deprecated.
See help("Deprecated")
```

```
In [24]: head( lm_coefs, 30 )
```

A df[,2]: 30 Œ 2

row	value
<chr>	<dbl>
(Intercept)	-18.37737222
MarFemaleDivorcees	-5.11554208
CostDivIncLT25	3.24460585
CostDivIncLT15	2.86530431
AncSwiss	0.98236846
BadPlumbing	0.80539507
LessEq2Rooms	0.54193077
BadKitchen	0.32900049
JobAgriculture	0.32006860
WorkClassSelf	0.19349322
IncRatioM2F	-0.12500977
VehicGT1	0.09359990
SchNurs	-0.08918499
EduHSDip	0.08194796
EduColNoDeg	-0.07702541
IncSupSec	0.07538104
NativityUS	0.07525924
CommuteAvgTravTime	0.06972501
IndConstruction	0.06312009
CommuteAtHome	0.05860763
BuiltAfter90	0.05768235
NativityOutUS	-0.05674349
IncRetirement	-0.05583241
MarWidowedFemales	-0.05569634
Resid5Yrs	0.05238722
AncItalian	-0.05140025
MarWidow	-0.05032510
SchKind	-0.04929637
MortgageLT700	0.04431094
AncCzech	0.04281635

8 Use best lambda for prediction

- For functions like `predict()`, arg s is where you input lambda

```
In [25]: dim( train_data )
```

1. 805 2. 245

```
In [26]: lm_ypred <- predict(
             glmnet_cv_result, test_data, s="lambda.min" ) %>%
             as.numeric # convert predictions to R vector from R matrix
```

```
In [27]: length(lm_ypred)
```

201

```
In [28]: all_pred_results <- tibble( y=test_data$Penetration, glmnet_linear=lm_ypred )
```

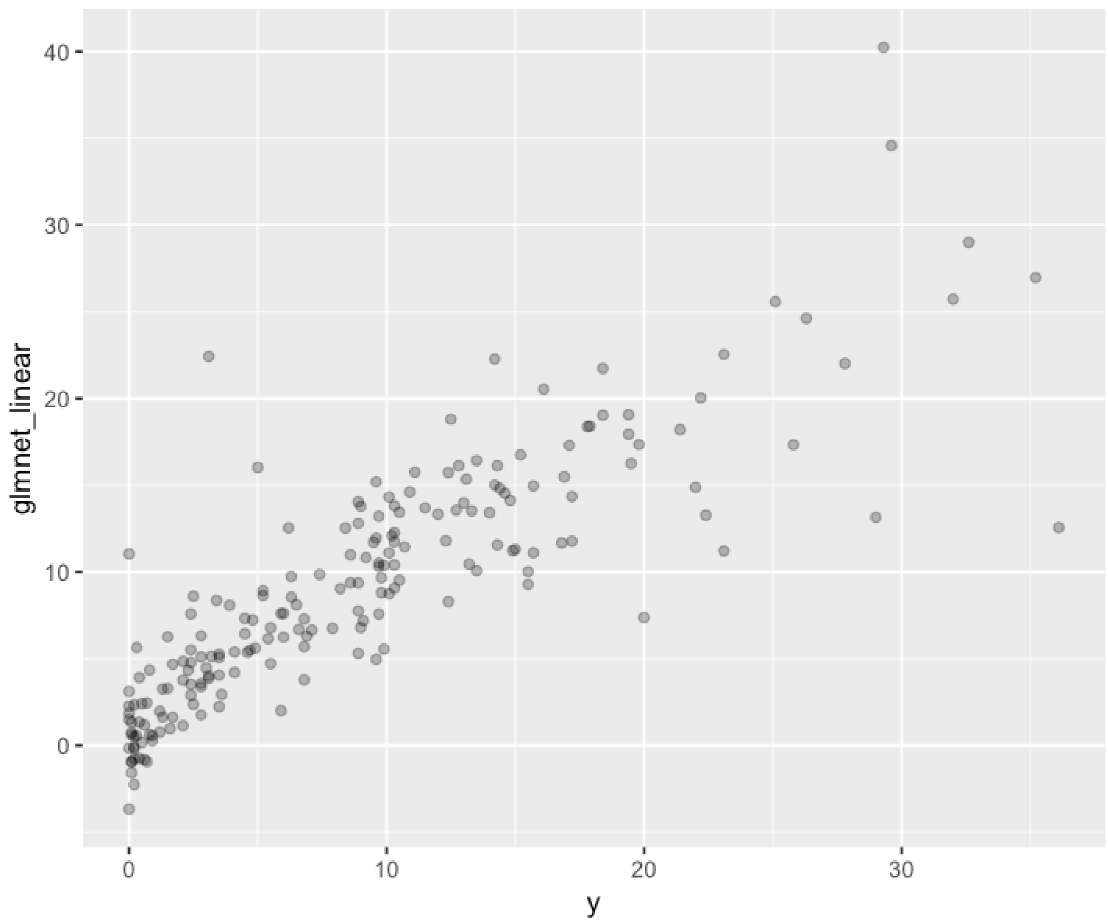
8.1 Model achieves training R-squared of 0.72

```
In [29]: summary( lm( glmnet_linear ~ y, all_pred_results) )


Call:
lm(formula = glmnet_linear ~ y, data = all_pred_results)

Residuals:
     Min       1Q   Median       3Q      Max
-17.4435  -2.0476  -0.1604   1.7866  17.6349

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.41597    0.39526   6.112 5.09e-09 ***
y            0.76420    0.03323  23.000  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.717 on 199 degrees of freedom
Multiple R-squared:  0.7266,Adjusted R-squared:  0.7253
F-statistic:   529 on 1 and 199 DF,  p-value: < 2.2e-16
```

8.2 Make plot of predicted vs actual

- Note negative predictions!!!

```
In [30]: ggplot( all_pred_results, aes( y, glmnet_linear ) ) +
             geom_point( alpha=0.3 )
```

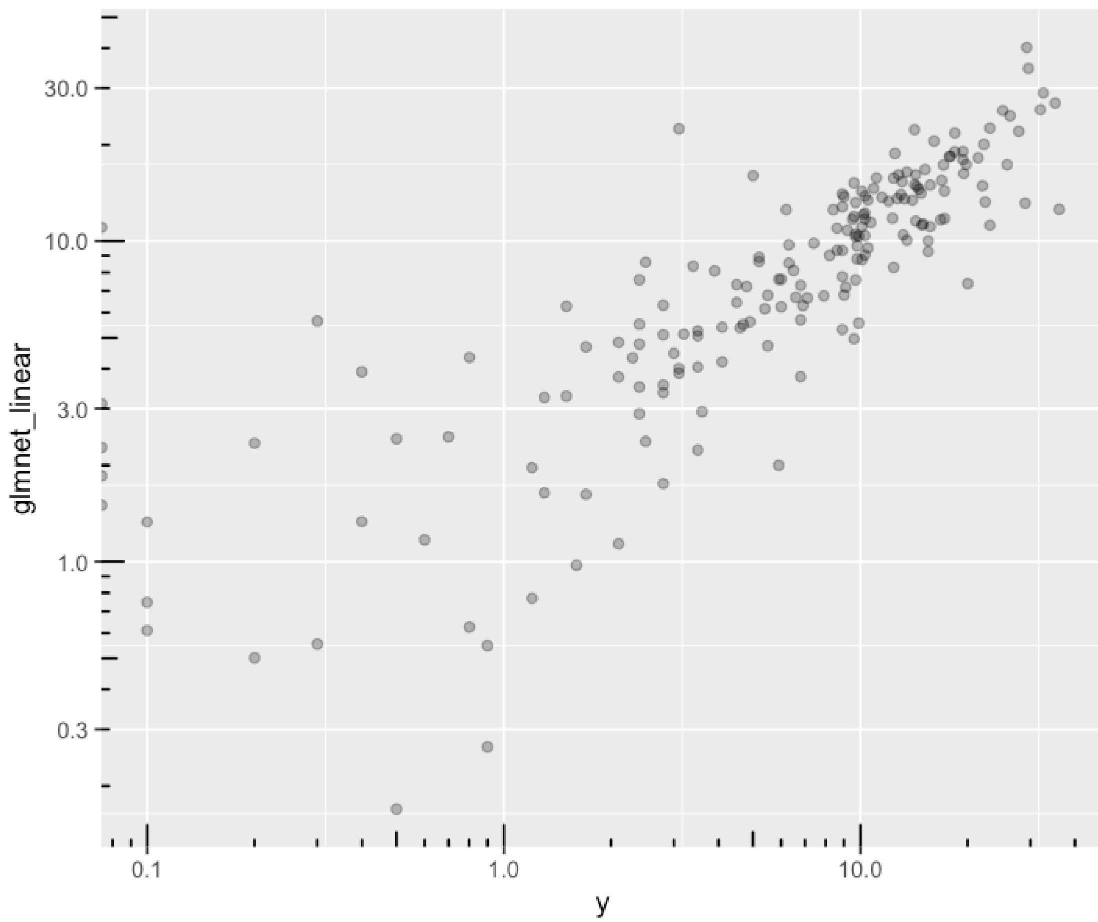


8.3 Make log-log plot of predicted vs actual

- ggplot throws warning messages because it can't handle log transformations of negative predictions - it drops them from the figure.

```
In [31]: ggplot( all_pred_results, aes( y, glmnet_linear ) ) +
             geom_point( alpha=0.3 ) +
             scale_x_continuous(trans = 'log10') +
             scale_y_continuous(trans = 'log10') +
             annotation_logticks()
```

```
Warning message:
Transformation introduced infinite values in continuous x-axisWarning message in self$trans$tra
NaNs producedWarning message:
Transformation introduced infinite values in continuous y-axisWarning message:
Removed 12 rows containing missing values (geom_point).
```



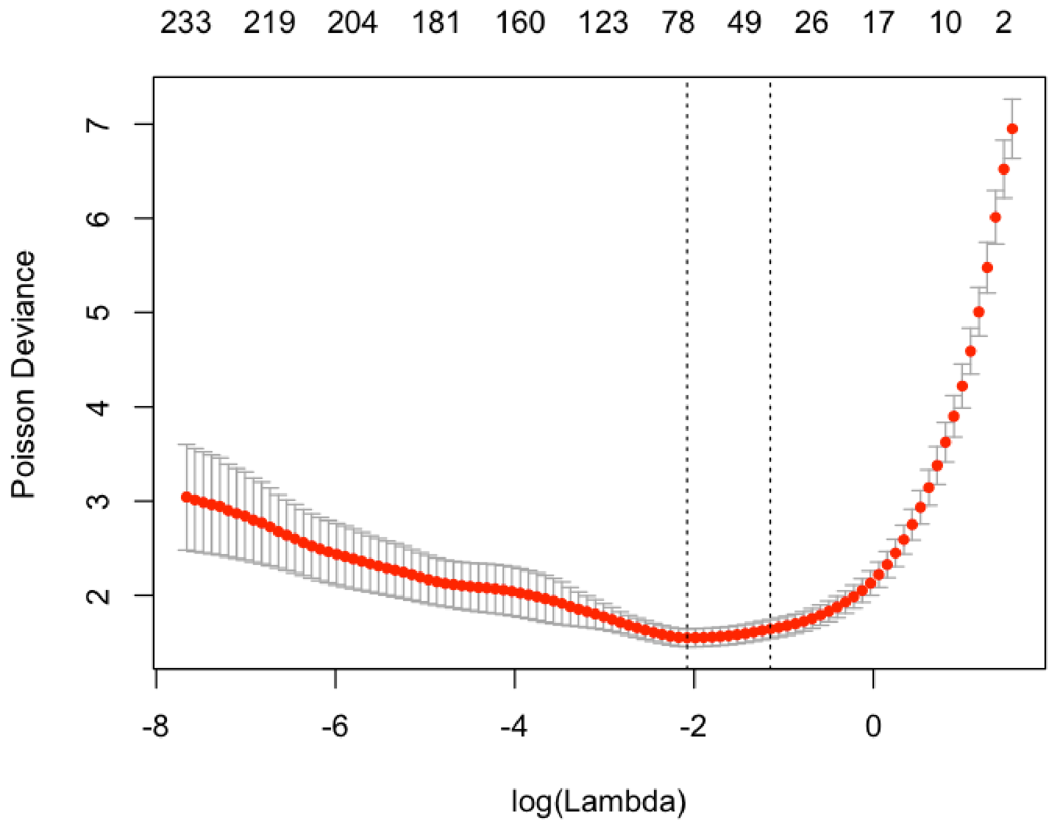
9 Slightly better: Generalized Linear model

- Model the outcome variable `Penetration` as having a poisson distribution
- Use cross validation to find optimal lambda, then make predictions with that lambda.
- GLM offsets - advanced topic not covered here

In [32]:
```
tic()
glmnet_cv_result2 <- cv.glmnet( Penetration ~ ., data=train_data, family='poisson' )
toc()
```

```
12.1 sec elapsed
```

In [33]:
```
plot( glmnet_cv_result2 )
```




```
In [34]: lm2_ypred <- predict(
             glmnet_cv_result2, test_data, s="lambda.min", type = "response" ) %>%
             as.numeric # convert to R vector from R matrix

In [35]: length( lm2_ypred )
```

201

```
In [36]: all_pred_results <- all_pred_results %>%
             mutate( glmnet_poisson = lm2_ypred )
```

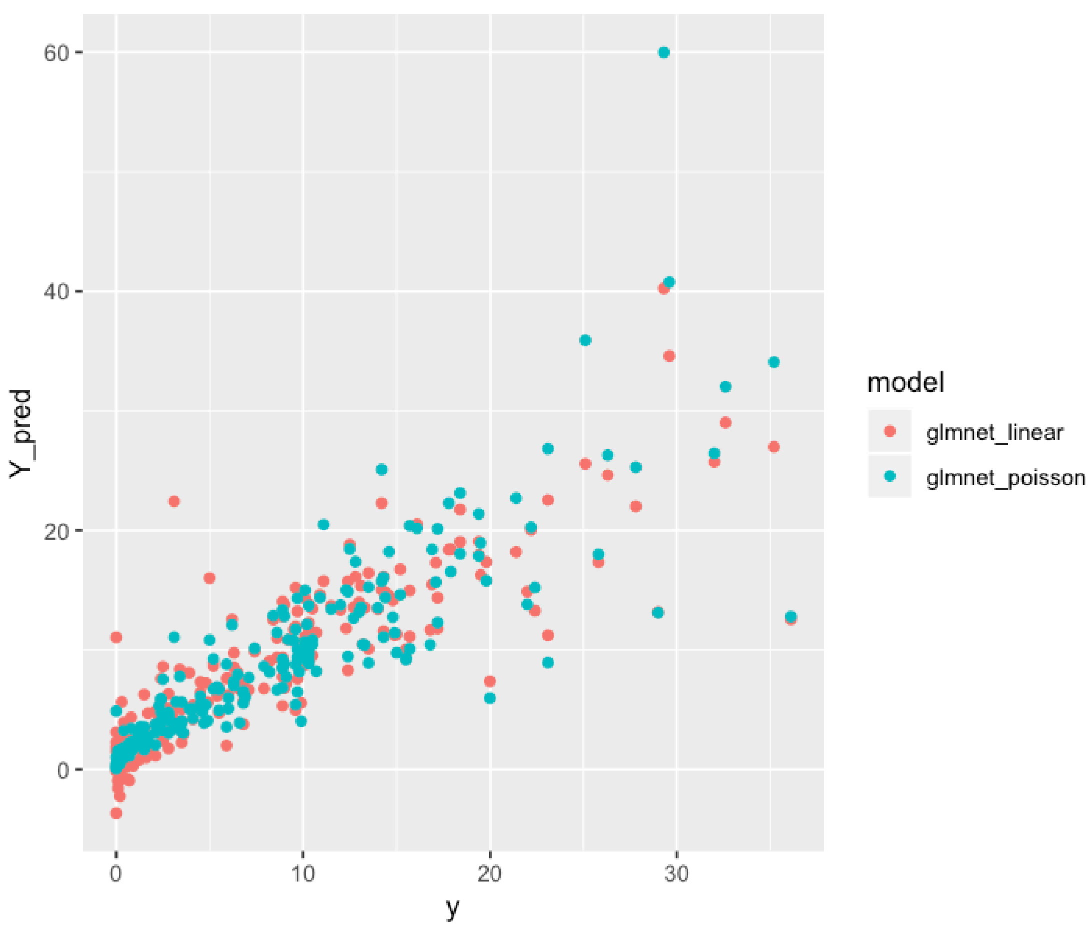
9.1 Poisson-style glmnet model R-squared is comparable to linear-style glmnet model

```
In [37]: summary( lm( glmnet_poisson ~ y, all_pred_results) )
```

```
Call:
lm(formula = glmnet_poisson ~ y, data = all_pred_results)
```

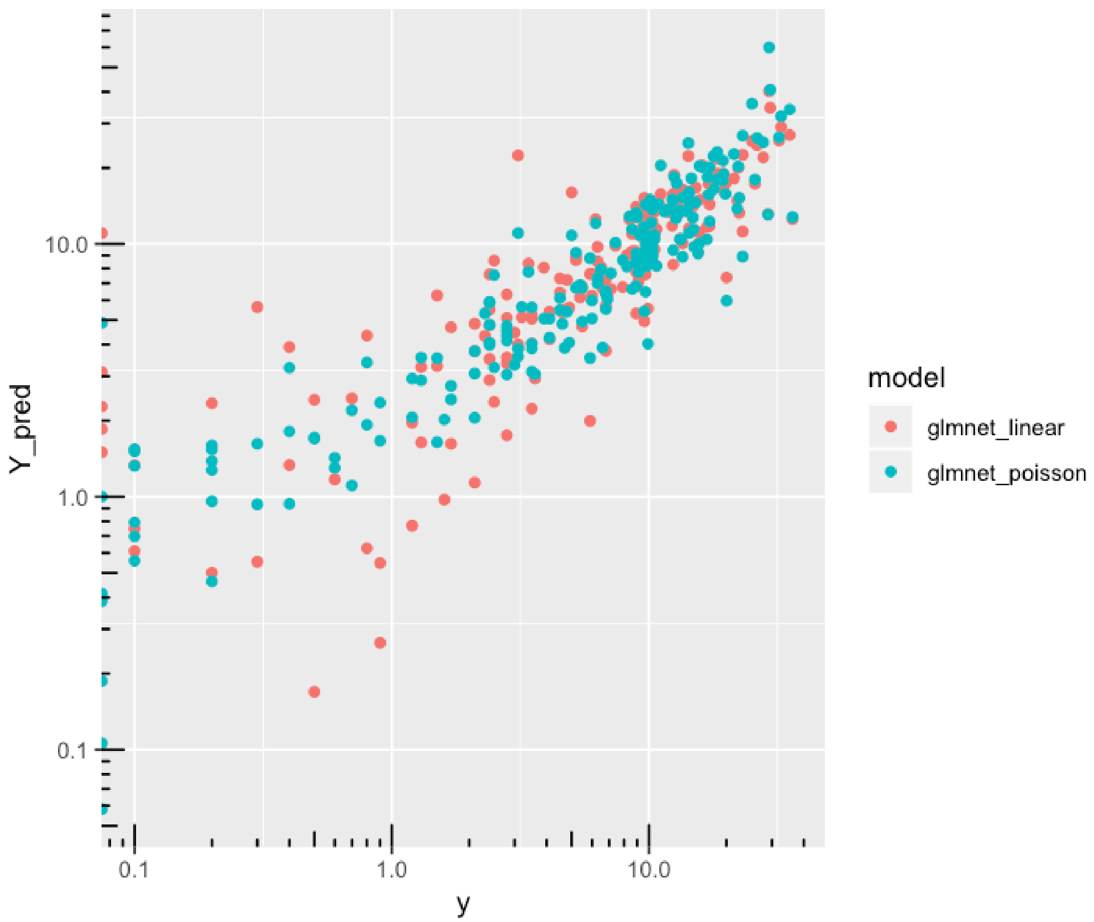
```
Residuals:
    Min      1Q  Median      3Q     Max
-20.227  -1.217  -0.325   1.229  32.886

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.63093    0.45776   3.563 0.000459 ***
y            0.86952    0.03848  22.596  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 4.305 on 199 degrees of freedom
Multiple R-squared:  0.7196,Adjusted R-squared:  0.7182
F-statistic: 510.6 on 1 and 199 DF,  p-value: < 2.2e-16
```

```
In [38]: all_pred_results %>%
            gather( key='model', value='Y_pred', -y ) %>%
            ggplot( aes( y, Y_pred, color=model) ) + geom_point()
```



```
In [39]: all_pred_results %>%
            gather( key='model', value='Y_pred', -y ) %>%
            ggplot( aes( y, Y_pred, color=model) ) + geom_point() +
            scale_x_continuous(trans = 'log10') +
            scale_y_continuous(trans = 'log10') +
            annotation_logticks()
```

```
Warning message:
Transformation introduced infinite values in continuous x-axisWarning message in self$trans$tra
NaNs producedWarning message:
Transformation introduced infinite values in continuous y-axisWarning message:
Removed 12 rows containing missing values (geom_point).
```



10 Use glmnet coefs for variable selection

```
In [40]: # remove row marked '(Intercept)'
         interesting_linear_vars <- c( setdiff( lm_coefs[1:15,'row'],  '(Intercept)'), 'Penetra
```

```
In [41]: interesting_linear_vars
```

1. 'MarFemaleDivorcees' 2. 'CostDivIncLT25' 3. 'CostDivIncLT15' 4. 'AncSwiss' 5. 'BadPlumbing' 6. 'LessEq2Rooms' 7. 'BadKitchen' 8. 'JobAgriculture' 9. 'WorkClassSelf' 10. 'IncRatioM2F' 11. 'VehicGT1' 12. 'SchNurs' 13. 'EduHSDip' 14. 'EduColNoDeg' 15. 'Penetration'

```
In [42]: train_data_subset <- train_data %>% select( interesting_linear_vars )
```

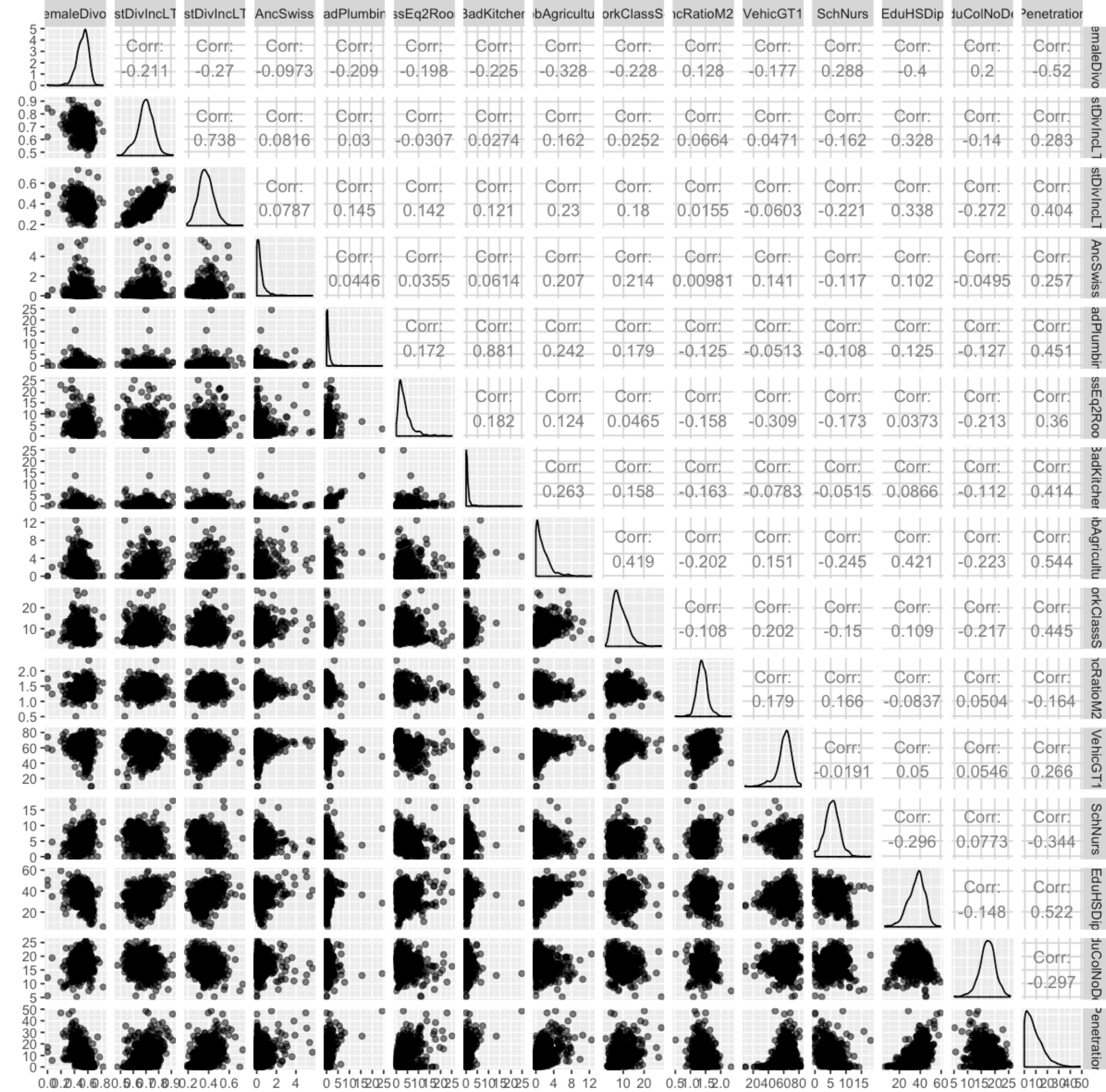
10.1 Exploratory: Univariate and Bivariate dists

```
In [43]: skim_to_wide( train_data_subset )  %>% select( -type, -n )
```

A tibble: 15 Œ 11

variable <chr>	missing <chr>	complete <chr>	mean <chr>	sd <chr>	p0 <chr>	p25 <chr>	p50 <chr>	p7 <c
AncSwiss	0	805	0.42	0.62	0	0.1	0.2	0.5
BadKitchen	0	805	0.68	1.3	0	0.1	0.4	0.9
BadPlumbing	0	805	0.74	1.33	0	0.2	0.4	0.9
CostDivIncLT15	0	805	0.37	0.075	0.19	0.33	0.37	0.4
CostDivIncLT25	0	805	0.7	0.063	0.47	0.66	0.7	0.7
EduColNoDeg	0	805	17.24	2.92	5.2	15.6	17.4	19
EduHSDip	0	805	36.23	7.89	6	31.4	37.2	41
IncRatioM2F	0	805	1.39	0.18	0.5	1.28	1.38	1.5
JobAgriculture	0	805	1.54	1.66	0	0.3	1.1	2.2
LessEq2Rooms	0	805	3.84	3.34	0	1.7	3	4.8
MarFemaleDivorcees	0	805	0.52	0.089	0	0.47	0.53	0.5
Penetration	0	805	8.66	7.9	0	2.5	6.7	13
SchNurs	0	805	5.58	2.36	0	4.1	5.6	7
VehicGT1	0	805	60.93	9.87	9.6	56.8	62.3	67
WorkClassSelf	0	805	8.61	3.65	1.6	6	7.9	10

```
In [44]: options(repr.plot.width=10, repr.plot.height=10)
         ggpairs( train_data_subset, aes( alpha=0.001 ) )
```



11 Regular Linear Model

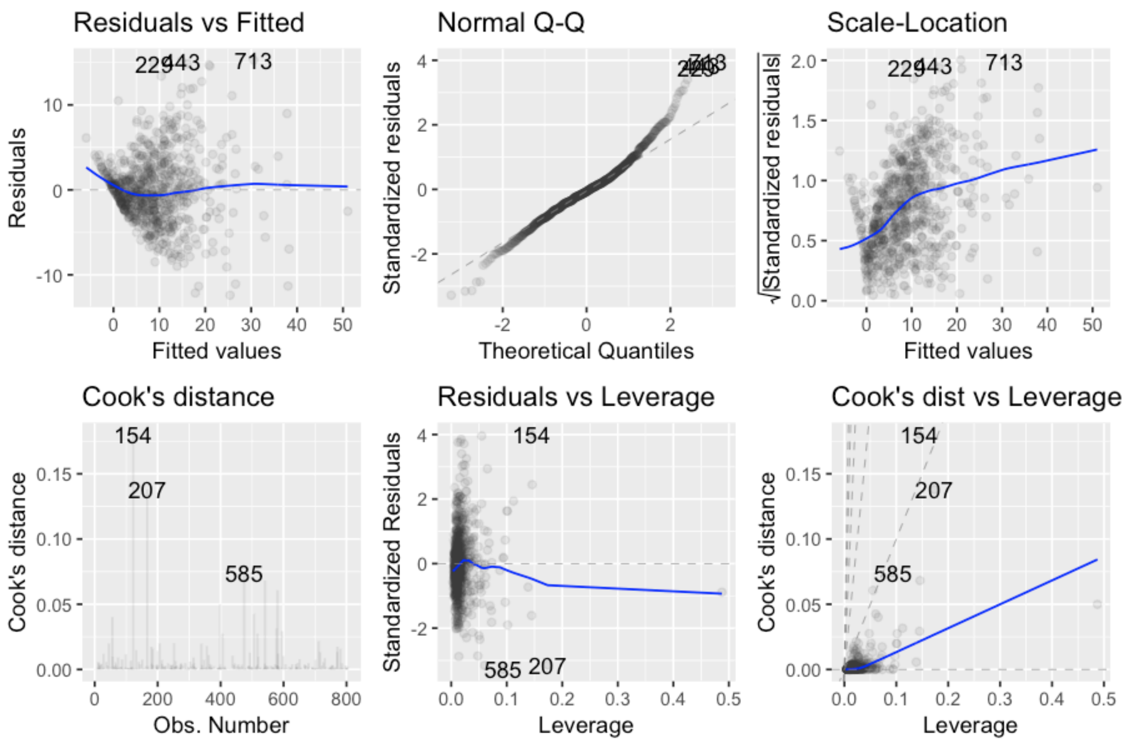
```
In [45]: model0 <- lm( Penetration ~ ., data=train_data_subset )

In [46]: summary( model0 )
```

```
Call:
lm(formula = Penetration ~ ., data = train_data_subset)

Residuals:
     Min       1Q   Median       3Q      Max
-12.3993  -2.3369  -0.2804   1.9329  15.2693
```

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)        -22.52440    2.93277  -7.680 4.70e-14 ***
MarFemaleDivorcees  -7.50162    1.89528  -3.958 8.24e-05 ***
CostDivIncLT25       5.32176    3.48363   1.528 0.127000    
CostDivIncLT15      11.31104    3.06880   3.686 0.000244 ***
AncSwiss             0.98030    0.23534   4.165 3.45e-05 ***
BadPlumbing          1.26269    0.22902   5.514 4.76e-08 ***
LessEq2Rooms         0.74801    0.04796  15.597  < 2e-16 ***
BadKitchen           0.38757    0.23727   1.633 0.102774    
JobAgriculture       0.41190    0.10843   3.799 0.000157 ***
WorkClassSelf        0.37321    0.04526   8.246 6.81e-16 ***
IncRatioM2F         -2.49474    0.85991  -2.901 0.003821 ** 
VehicGT1             0.24610    0.01651  14.906  < 2e-16 ***
SchNurs             -0.15866    0.06578  -2.412 0.016098 *  
EduHSDip             0.29024    0.02195  13.223  < 2e-16 ***
EduColNoDeg         -0.13671    0.05213  -2.623 0.008895 ** 
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.969 on 790 degrees of freedom
Multiple R-squared:  0.7519,Adjusted R-squared:  0.7475 
F-statistic:   171 on 14 and 790 DF,  p-value: < 2.2e-16
```

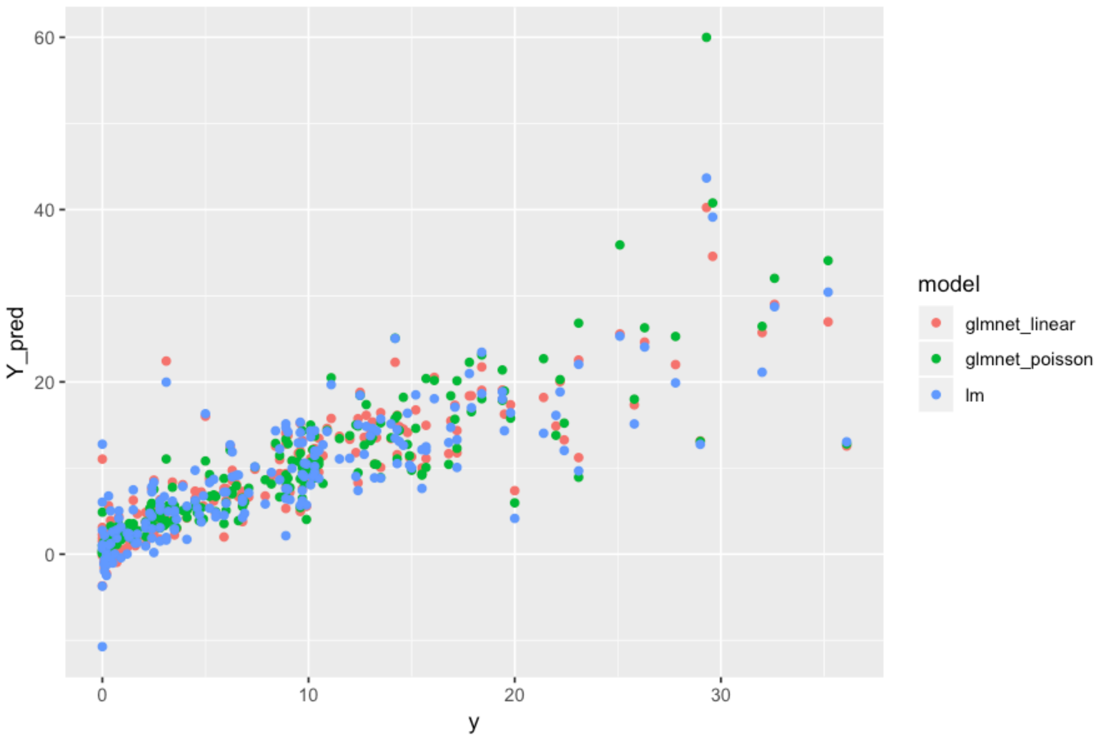
```
In [47]: options( repr.plot.width=7.5, repr.plot.height=5 )
```

```
In [48]: autoplot( model0, which = 1:6, ncol=3, alpha=0.1)
```




```
In [49]: all_pred_results <- all_pred_results %>%
             mutate( lm = predict( model0, test_data ) )

In [50]: all_pred_results %>%
             gather( key='model', value='Y_pred', -y ) %>%
             ggplot( aes( y, Y_pred, color=model) ) + geom_point() #+
             #scale_x_continuous(trans = 'log10') +
             #scale_y_continuous(trans = 'log10') +
             #annotation_logticks()
```

```
Warning message:
attributes are not identical across measure variables;
they will be dropped
```



12 GLM model

```
In [51]: glm_coefs <- glmnet_cv_result2 %>%
             coef( s = "lambda.min" ) %>%    # Get the betas from the best model
             tidy %>%                        # Put betas into a data.frame
             select( -column ) %>%           # prune unimportant column named "column"
             arrange( desc( abs( value ) ) ) # Sort by absolute value
```

```
Warning message:
'tidy.dgCMatrix' is deprecated.
See help("Deprecated")Warning message:
'tidy.dgTMatrix' is deprecated.
See help("Deprecated")
```

```
In [52]: glm_coefs %>% head(15)
```

A df[,2]: 15 Œ 2

row <chr>	value <dbl>
MarFemaleDivorcees	-0.38459635
CostDivIncLT25	0.38011921
LabFCivilEmployed	0.25292790
(Intercept)	0.21780932
AncSubSah	-0.13735906
CostDivIncLT15	0.13313636
AncSwiss	0.04909032
BadKitchen	0.02717750
AncItalian	-0.02633768
HouseMultiFamily	-0.02609654
AncCzech	0.02494139
BadPlumbing	0.02413368
WorkClassSelf	0.02033181
JobAgriculture	0.01565186
IncSupSec	0.01512092

In [53]:
```
# remove row marked '(Intercept)'
interesting_poisson_vars <- c( setdiff( glm_coefs[1:15,'row'],  '(Intercept)'), 'Pene
```

In [54]:
```
data_subset <- train_data %>% select( interesting_poisson_vars )
```

In [55]:
```
model1 <- glm( Penetration ~ ., data=data_subset, family='quasipoisson' )
```

In [56]:
```
summary( model1 )
```

```
Call:
glm(formula = Penetration ~ ., family = "quasipoisson", data = data_subset)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.0784  -1.0049  -0.2477   0.6750   4.6136

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         2.499317   0.457565   5.462 6.30e-08 ***
MarFemaleDivorcees -1.028500   0.178225  -5.771 1.13e-08 ***
CostDivIncLT25      0.848191   0.362827   2.338 0.019650 *
LabFCivilEmployed  -0.207178   0.420219  -0.493 0.622133
AncSubSah          -0.357826   0.117860  -3.036 0.002476 **
CostDivIncLT15     -0.004742   0.315302  -0.015 0.988005
AncSwiss            0.085180   0.018595   4.581 5.38e-06 ***
BadKitchen         -0.009841   0.019379  -0.508 0.611718
AncItalian         -0.043466   0.003932 -11.053  < 2e-16 ***
HouseMultiFamily   -0.044923   0.002607 -17.233  < 2e-16 ***
AncCzech            0.099778   0.029782   3.350 0.000846 ***
BadPlumbing         0.045596   0.019257   2.368 0.018138 *
```

```
WorkClassSelf       0.023411   0.004522   5.177 2.86e-07 ***
JobAgriculture      0.053187   0.009138   5.820 8.53e-09 ***
IncSupSec           0.041014   0.007111   5.768 1.15e-08 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for quasipoisson family taken to be 1.740372)

    Null deviance: 5592.2  on 804  degrees of freedom
Residual deviance: 1424.6  on 790  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

In [57]:
```
glm_preds <- as.numeric( predict( model1, test_data, type='response' ) )
```

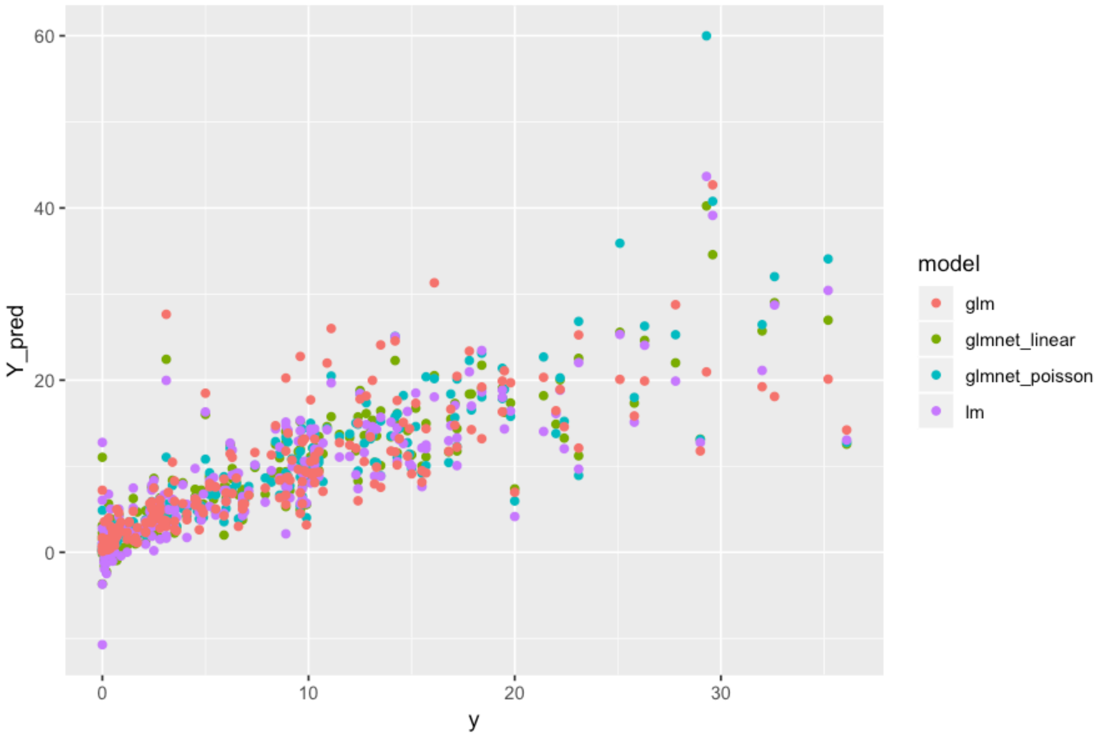
In [58]:
```
all_pred_results <- all_pred_results %>%
    mutate( glm =  glm_preds )
```

In [59]:
```
summary( lm( glm ~ y, all_pred_results ) )
```

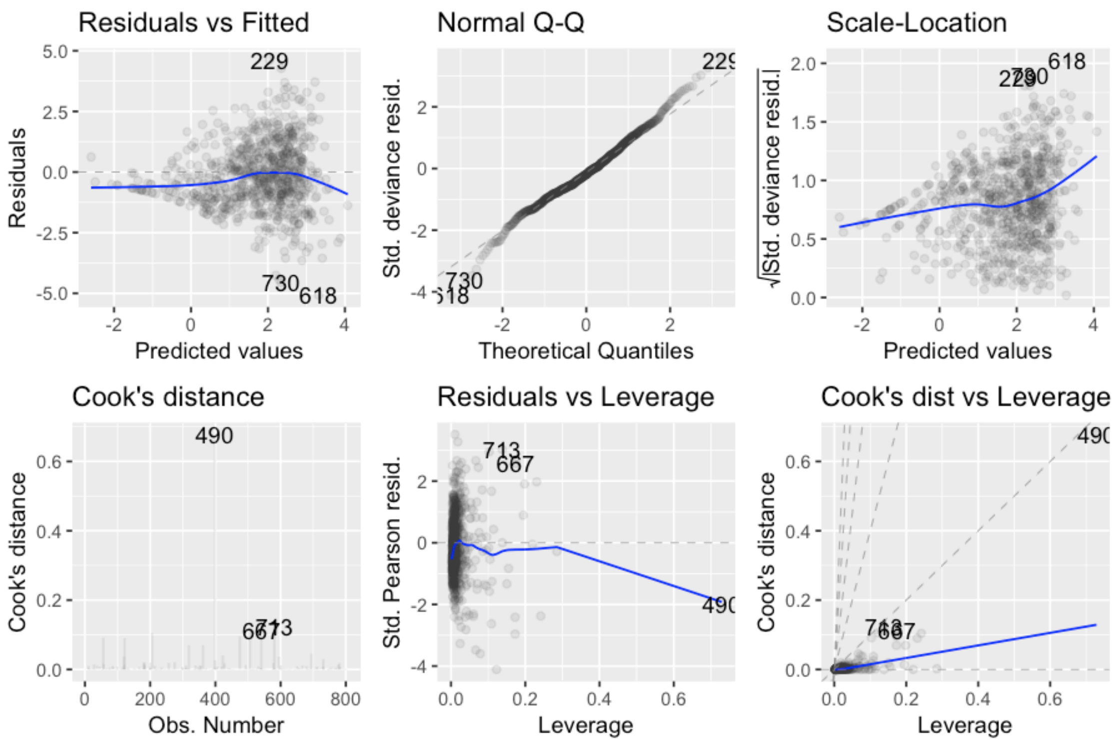
```
Call:
lm(formula = glm ~ y, data = all_pred_results)

Residuals:
     Min       1Q   Median       3Q      Max
-13.8162  -2.4198  -0.9994   1.2441  22.4002

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.10600    0.49328   6.297 1.91e-09 ***
y            0.69032    0.04147  16.648  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 4.639 on 199 degrees of freedom
Multiple R-squared:  0.5821,Adjusted R-squared:   0.58
F-statistic: 277.1 on 1 and 199 DF,  p-value: < 2.2e-16
```

In [60]:
```
all_pred_results %>%
    gather( key='model', value='Y_pred', -y ) %>%
    ggplot( aes( y, Y_pred, color=model) ) + geom_point() #+
    #scale_x_continuous(trans = 'log10') +
    #scale_y_continuous(trans = 'log10') +
    #annotation_logticks()
```

```
Warning message:
attributes are not identical across measure variables;
they will be dropped
```




```
In [61]: autoplot( model1, which = 1:6, ncol=3, alpha=0.1)
```



12.1 What about log transforming the skewed vars?

```
In [ ]: log_transform_vars <- c( "MortageLT500", "CommuteAtHome", 'BadPlumbing', "BoatRVVan",

In [ ]: transformed_data_subset <- data_subset %>%
            mutate_at( log_transform_vars, ~ log( . + 1) )

In [ ]: skim_to_wide( transformed_data_subset )

In [ ]: skim_to_wide( data_subset )

In [ ]: model2 <- glm( Penetration ~ ., data=transformed_data_subset, family='quasipoisson' )

In [ ]: summary( model2 )

In [ ]: autoplot( model2, which = 1:6, ncol=3)

In [ ]: anova( model1, model2)
```