432 Week 4 Slides

github.com/THOMASELOVE/2020-432

2020-02-04 & 02-06

This Week's Agenda

- County Health Rankings Data
- Pre-processing of data
- Investigating an outcome transformation
- Model (Variable) Selection
  - Why Stepwise Regression is Terrible
  - How to do "Best Subsets" Linear Regression
- Cross-Validation
  - Validation Set approach
  - K-fold Cross-Validation
- and perhaps some more...

Setup

```r
library(here); library(magrittr); library(janitor)
library(patchwork); library(naniar)
library(knitr); library(mosaic); library(skimr)

library(GGally)
library(car)
library(leaps)
library(caret)
library(modelr)

library(broom)
library(tidyverse)

theme_set(theme_bw())
```

US County Health Rankings, 2017

US County Health Rankings Data

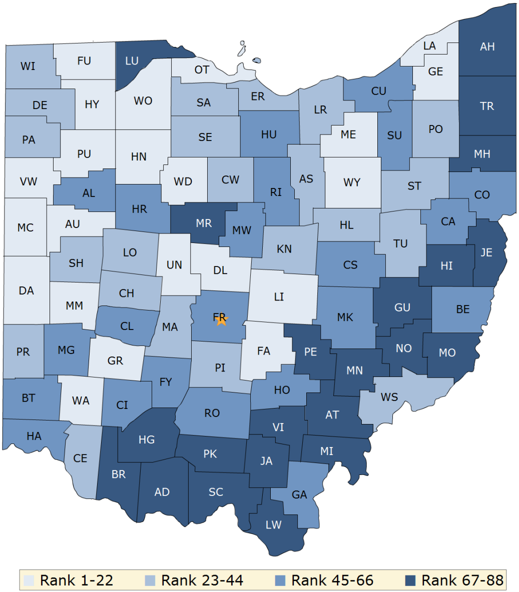
- We'll use the 2017 data in our examples, to leave more recent data available for you in Projects and future work.
- Data source is this link at https://www.countyhealthrankings.org.

Next two slides show Ohio counties in 2017 and their rankings on:

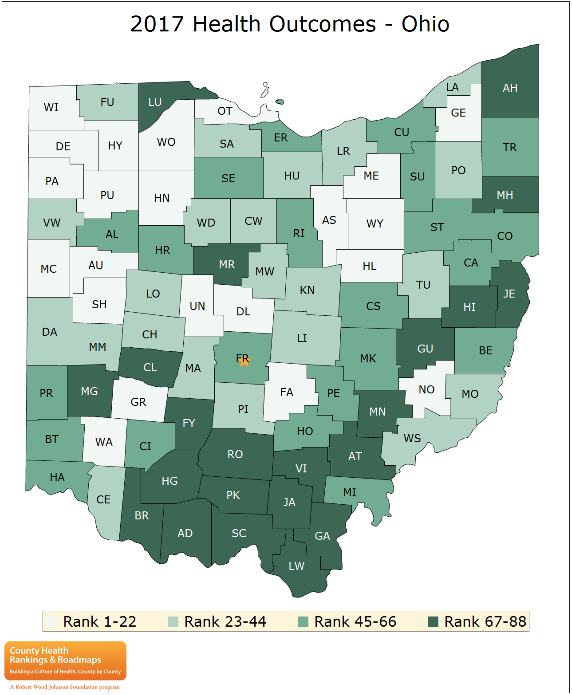
- Health Factors: weighted scores for health behaviors, clinical care, social and economic factors, and the physical environment (Cuyahoga ranked 56th out of the 88 Ohio counties)
- Health Outcomes: equal weighting of length and quality of life (Cuyahoga ranked 65th)

In each slide, lighter shades indicate better performance.

2017 Health Factors - Ohio



County Health Rankings & Roadmaps
Building a Culture of Health, County by County
A Robert Wood Johnson Foundation program


2017 Health Outcomes - Ohio
WI
FU
LU
OT
LA
AH
GE
DE
HY
WO
SA
ER
CU
LR
TR
PA
PU
HN
SE
HU
ME
SU
PO
MH
VW
AL
WD
CW
RI
AS
WY
ST
CO
MC
AU
HR
MR
MW
HL
CA
SH
LO
UN
DL
KN
TU
JE
CS
HI
DA
MM
CH
LI
GU
BE
CL
MA
FR
MK
PR
MG
GR
FA
PE
NO
MO
PI
MN
FY
HO
WS
BT
WA
CI
RO
AT
VI
HA
HG
PK
MI
CE
JA
BR
AD
SC
GA
LW
Rank 1-22
Rank 23-44
Rank 45-66
Rank 67-88
County Health Rankings & Roadmaps
Building a Culture of Health, County by County
A Robert Wood Johnson Foundation program

Data Ingest and Cleanup

```
count_2017 <-
    read_csv(here("data/countyhealthrankings_2017.csv")) %>%
    clean_names() %>%
    type.convert() %>%
    mutate(fips = as.character(fips),
           state = as.character(state),
           county = as.character(county))

dim(count_2017)
```

```
[1] 3136   34
```

Codebook (2017 County Health Rankings), I

Variable	Description
`fips`	FIPS code (5 digits: 1-2: state, 3-5: county)
`state`	State Name
`county`	County Name
`yrs_lost_rate`	Years of potential life lost before age 75 per 100,000 population (age-adjusted, 2012-14)
`population`	County population in 1000s, Census Population Estimates, 2015
`not_english`	% of county residents preferring language besides English (ACS 2010)
`hh_income`	Median Household Income in $1000s (Small Area Income & Poverty Estimates 2015)
`income_ratio`	Ratio of household income at the 80th percentile to income at the 20th percentile (ACS 2011-15)
`health_costs`	Average per-capita health care costs (Dartmouth Atlas of Health Care 2014)

Codebook (2017 County Health Rankings), II

Variable	Description
social_assoc	Membership associations per 10,000 population (County Business Partners 2014)
pct_smokers	% of adults who currently smoke (2015 BRFSS)
food_envir	Food environment index (0 = worst, 10 = best) (USDA Map the Meal 2014)
housing_prob	% of households with at least 1 of 4 severe problems (Comp. Housing Affordability Strategy 2009-13)
drive_alone	% of the workforce that drives alone to work (ACS 2011-15)
rural_cat	from % rural (0-20: Urban, 20.01-50: Suburban, 50.01+: Rural; Census 2015)
race_cat	from 100 - % Non-Hispanic White: ($>$ 20: High, 10-20: Middle, 5-10: Low, $<$ 5: Very Low; Census 2015)

Selecting Our Sample of Variables, Observations

```
ourbatch <- count_2017 %>%
    filter(complete.cases(yrs_lost_rate),
           state %in% c("Illinois", "Indiana", "Michigan",
                 "Minnesota", "Ohio", "Wisconsin",
                 "Iowa", "Missouri")) %>%
    select(fips, state, county, yrs_lost_rate,
           population, not_english, hh_income,
           income_ratio, health_costs, social_assoc,
           pct_smokers, food_envir, housing_prob,
           drive_alone, rural_cat, race_cat)

dim(ourbatch)
```

```
[1] 734  16
```

Missing Data?

```
pct_miss(ourbatch)
```

```
[1] 0
```

```
ourbatch %>% select(fips, state, county) %>%
  summarize_all(list(n_distinct))
```

```
# A tibble: 1 x 3
   fips state county
  <int> <int>  <int>
1   734     8    486
```

1. Which states have the most/least counties?
2. Why do we have fewer county names than `fips` codes?

Counties in each state?

```
ourbatch %>% tabyl(state) %>% adorn_pct_formatting()
```

```
     state   n percent
  Illinois 102   13.9%
   Indiana  92   12.5%
      Iowa  99   13.5%
  Michigan  82   11.2%
 Minnesota  85   11.6%
  Missouri 114   15.5%
      Ohio  88   12.0%
 Wisconsin  72    9.8%
```

Across the US, the most common county name is Washington.

```r
ourbatch %>% count(county, sort = TRUE) %>% filter(n > 5)
```

```
# A tibble: 8 x 2
  county         n
  <chr>      <int>
1 Jackson        8
2 Crawford       7
3 Monroe         7
4 Washington     7
5 Cass           6
6 Clinton        6
7 Jefferson      6
8 Wayne          6
```

Basic Data Summaries

Available approaches include:

- summary
- mosaic package's inspect()
- skimr package's skim_without_charts()
- Hmisc package's describe()

but none of them fit well on a single slide.

```
skimmed_predictors <- ourbatch %>%
    select(-fips, -state, -county, -yrs_lost_rate) %>%
  skim_without_charts()
```

The Quantitative Predictors

- Any concerns regarding the ranges of these variables?

```
yank(skimmed_predictors, "numeric") %>%
  select(skim_variable, p0, p50, p100, mean, sd) %>%
  kable(digits = 1)
```

skim_variable	p0	p50	p100	mean	sd
population	3694.0	28793.0	5238216.0	83750.2	249815.3
not_english	0.0	0.5	11.7	0.8	1.1
hh_income	28.2	49.9	97.7	50.5	9.7
income_ratio	3.2	4.1	7.2	4.2	0.5
health_costs	6387.4	9114.0	13702.9	9179.4	1087.9
social_assoc	5.2	14.5	43.7	15.5	5.6
pct_smokers	12.2	17.3	35.7	17.9	2.9
food_envir	3.8	7.7	9.3	7.6	0.8
housing_prob	6.0	12.7	24.3	12.8	3.0
drive_alone	51.7	81.5	89.2	80.9	4.5

Cuyahoga County's Values

```
cuya <- ourbatch %>% filter(county == "Cuyahoga")
```

fips	state	county	yrs_lost_rate	population	not_english
39035	Ohio	Cuyahoga	7827.8	1255921	1.85

hh_income	income_ratio	health_costs	social_assoc	pct_smokers
45.51	5.63	10023.93	9.2	18.7

food_envir	housing_prob	drive_alone	rural_cat	race_cat
6.5	18.83	80.23	Urban	High

The Categorical Predictors

```
ourbatch %>% tabyl(rural_cat, race_cat)
```

```
rural_cat High Low Middle Very Low
    Rural   16 179     63      196
 Suburban   19  87     84       10
    Urban   46   2     32        0
```

- There are two problems I see in this table.

Oops, better re-order that race_cat factor.

```r
ourbatch <- ourbatch %>%
    mutate(race_cat = fct_relevel(race_cat,
                                    "High", "Middle", "Low"))

ourbatch %>% tabyl(rural_cat, race_cat)
```

```
 rural_cat High Middle Low Very Low
     Rural   16     63 179      196
  Suburban   19     84  87       10
     Urban   46     32   2        0
```

- Now, would an interaction between rural_cat and race_cat be helpful in building models?

Smallest Counties by Population

```
ourbatch %>% arrange(population) %>%
    select(state, county, population, yrs_lost_rate) %>%
    slice(1:10) %>% kable(digits = 3)
```

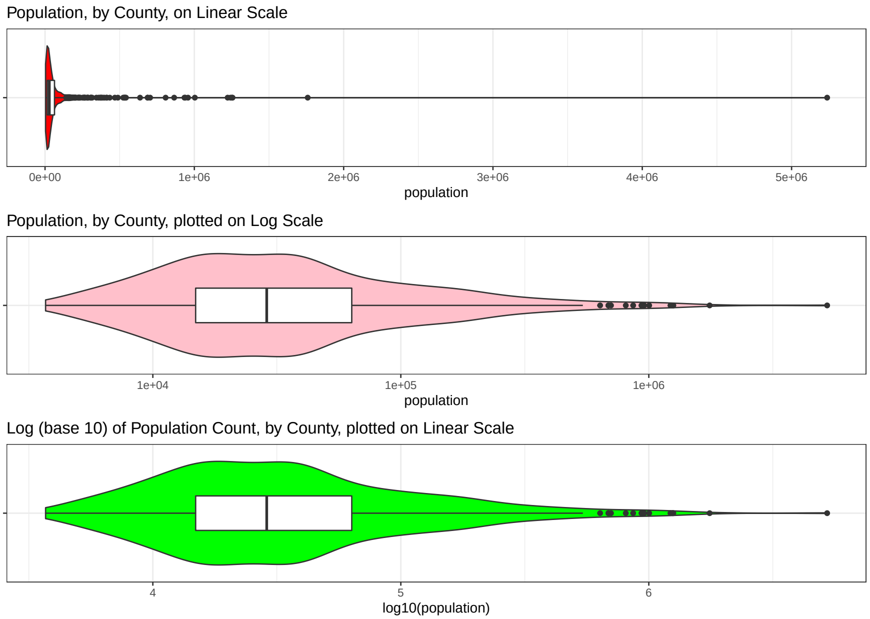
state	county	population	yrs_lost_rate
Missouri	Mercer	3694	6977.9
Iowa	Adams	3796	6717.5
Missouri	Knox	3910	10133.0
Minnesota	Red Lake	4055	3980.0
Illinois	Hardin	4135	13162.8
Illinois	Pope	4226	10811.1
Minnesota	Kittson	4424	6937.3
Missouri	Schuyler	4436	6670.6
Wisconsin	Florence	4464	6281.9
Missouri	Holt	4484	7290.9

Largest Counties by Population

```r
ourbatch %>% arrange(desc(population)) %>%
    select(state, county, population, yrs_lost_rate) %>%
    slice(1:10) %>% kable(digits = 3)
```

state	county	population	yrs_lost_rate
Illinois	Cook	5238216	6572.4
Michigan	Wayne	1759335	9960.6
Ohio	Cuyahoga	1255921	7827.8
Ohio	Franklin	1251722	7484.5
Michigan	Oakland	1242304	5805.6
Minnesota	Hennepin	1223149	4933.1
Missouri	St. Louis	1003362	6750.1
Wisconsin	Milwaukee	957735	7976.7
Indiana	Marion	939020	8992.1
Illinois	DuPage	933736	4059.0

Distribution of Population (code to follow)


Population, by County, on Linear Scale

Population, by County, plotted on Log Scale

Log (base 10) of Population Count, by County, plotted on Linear Scale

Code for plots of population distribution (start)

```r
p1 <- ggplot(ourbatch, aes(x = "",
                           y = population)) +
    geom_violin(fill = "red") +
    geom_boxplot(width = 0.3) + coord_flip() +
    labs(title = "Population, by County, on Linear Scale",
         x = "")

p2 <- ggplot(ourbatch, aes(x = "",
                           y = population)) +
    geom_violin(fill = "pink") +
    geom_boxplot(width = 0.3) + coord_flip() +
    scale_y_log10() +
    labs(title = "Population, by County, plotted on Log Scale"
         x = "")
```

Code for plots of population distribution (finish)

```
p3 <- ggplot(ourbatch, aes(x = "",
                           y = log10(population))) +
    geom_violin(fill = "green") +
    geom_boxplot(width = 0.3) + coord_flip() +
    labs(title = "Log (base 10) of Population Count, by County
         x = "")

p1 / p2 / p3
```

I’m going to transform the population information

```
ourbatch <- ourbatch %>%
    mutate(log_pop = log10(population))

ourbatch %$% Hmisc::describe(log_pop)
```

```
log_pop
       n  missing distinct     Info     Mean      Gmd
     734        0      729        1    4.525   0.5612
     .05      .10      .25      .50      .75      .90
   3.807    3.947    4.173    4.459    4.802    5.228
     .95
   5.468

lowest : 3.567497 3.579326 3.592177 3.607991 3.616476
highest: 6.094228 6.097508 6.098962 6.245349 6.719183
```

Our "Model Selection" Problem

We want to build a model to effectively predict our outcome `yrs_lost_rate` using the 10 quantitative candidate predictors we've identified. (We'll tackle the categorical predictors separately, later.)

1. `log_pop`
2. `not_english`
3. `hh_income`
4. `income_ratio`
5. `health_costs`
6. `social_assoc`
7. `pct_smokers`
8. `food_envir`
9. `housing_prob`
10. `drive_alone`

We have 734 counties (observations) to use here.

Our outcome is Age-Adjusted Years Lost Rate

Code for plotting using patchwork

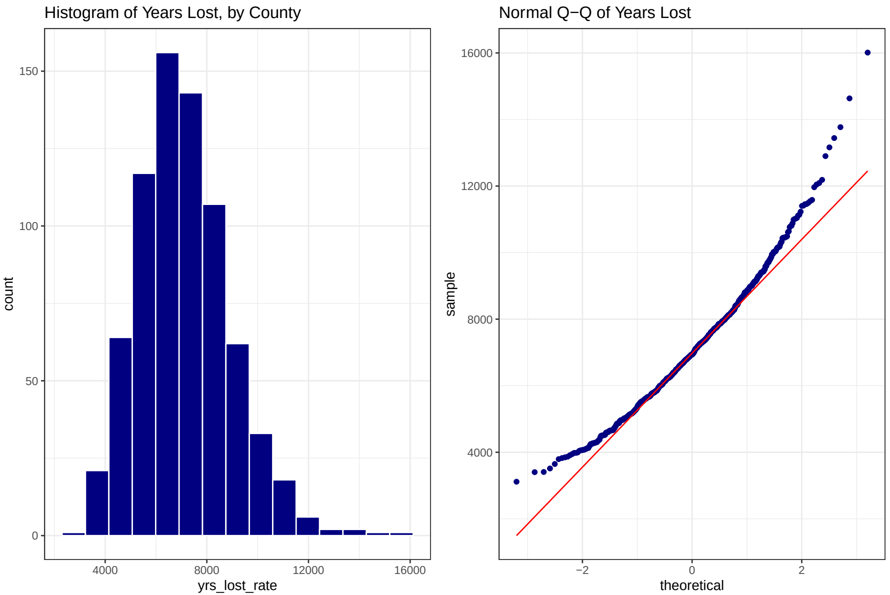
```
p1 <- ggplot(ourbatch, aes(yrs_lost_rate)) +
    geom_histogram(bins = 15,
                   fill = "navy", col = "white") +
    labs(title = "Histogram of Years Lost, by County")

p2 <- ggplot(ourbatch, aes(sample = yrs_lost_rate)) +
    geom_qq(col = "navy") + geom_qq_line(col = "red") +
    labs(title = "Normal Q-Q of Years Lost")

p1 + p2 +
    plot_annotation(title = "Age-Adjusted Years Lost before Ag
```

Our outcome is Age-Adjusted Years Lost Rate

Age−Adjusted Years Lost before Age 75 per 100,000 population



yrs_lost_rate: Ten Highest Counties

state	county	population	yrs_lost_rate
Missouri	Pemiscot	17482	16011.2
Wisconsin	Menominee	4573	14634.6
Indiana	Scott	23744	13770.0
Missouri	Dunklin	30895	13442.6
Illinois	Hardin	4135	13162.8
Missouri	Iron	10125	12899.5
Minnesota	Mahnomen	5457	12186.4
Ohio	Pike	28217	12091.3
Iowa	Monona	8979	12045.0
Indiana	Fayette	23434	11965.2

`yrs_lost_rate`: Ten Lowest Counties

state	county	population	yrs_lost_rate
Minnesota	Houston	18773	3115.1
Minnesota	Stevens	9796	3402.8
Minnesota	Carver	98741	3408.3
Minnesota	Scott	141660	3513.7
Iowa	Lyon	11745	3648.1
Minnesota	Marshall	9423	3794.8
Indiana	Hamilton	309697	3826.8
Iowa	Sioux	34937	3848.3
Iowa	Winneshiek	20709	3867.2
Iowa	Dallas	80133	3909.7

Hmisc::describe on `yrs_lost_rate`

```r
ourbatch %$% Hmisc::describe(yrs_lost_rate)
```

```
yrs_lost_rate
       n  missing distinct     Info     Mean      Gmd
     734        0      732        1     7124     2004
     .05      .10      .25      .50      .75      .90
    4505     4969     5821     6939     8129     9415
     .95
   10366

lowest :  3115.1  3402.8  3408.3  3513.7  3648.1
highest: 13162.8 13442.6 13770.0 14634.6 16011.2
```

A "Kitchen Sink" Model?

Predict yrs_lost_rate using the 12 candidate predictors.

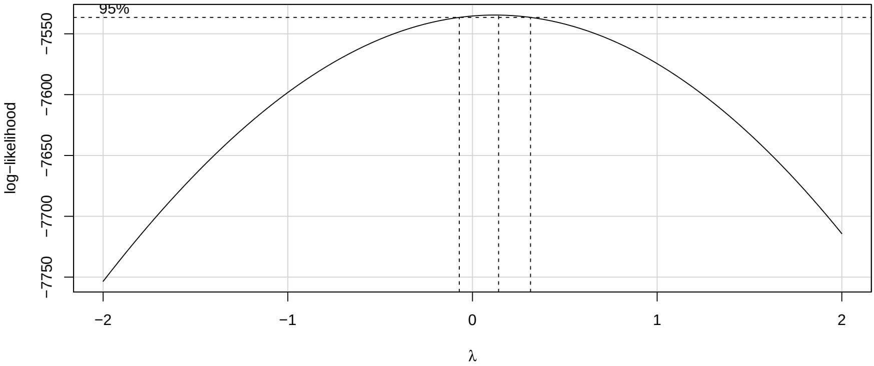
```
transformation_check <-
    lm(yrs_lost_rate ~
           log_pop + not_english + hh_income +
           income_ratio + health_costs + social_assoc +
           pct_smokers + food_envir + housing_prob +
           drive_alone + rural_cat + race_cat,
       data = ourbatch)

glance(transformation_check) %>%
  select(r.squared, adj.r.squared)
```

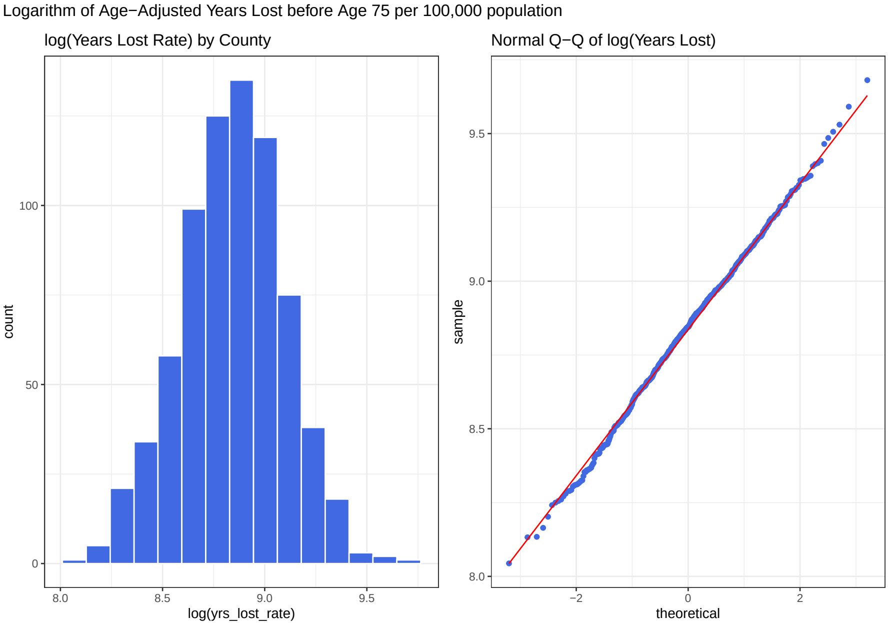
```
# A tibble: 1 x 2
  r.squared adj.r.squared
      <dbl>         <dbl>
1     0.618         0.610
```

Box-Cox plot: Outcome transformation?

```
boxCox(transformation_check)
```



Logarithm of our outcome rate?

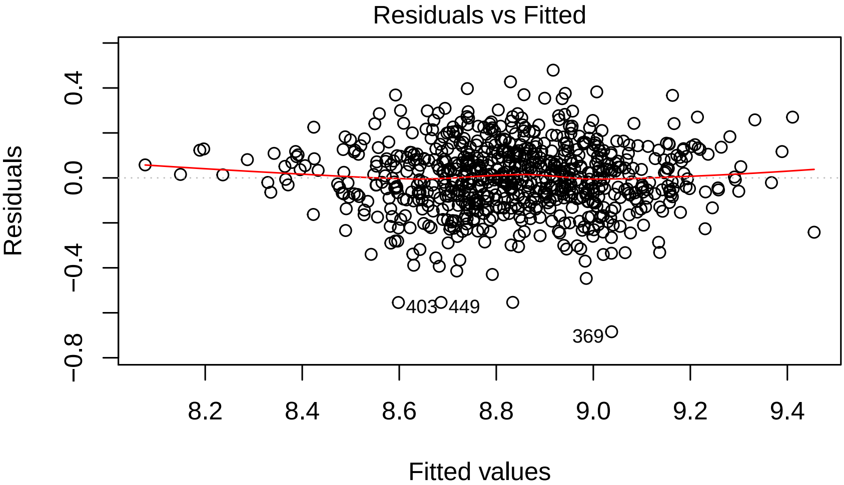


Kitchen Sink fit to log(yrs_lost_rate)

```
ks_log <-
    lm(log(yrs_lost_rate) ~
           log_pop + not_english + hh_income +
           income_ratio + health_costs + social_assoc +
           pct_smokers + food_envir + housing_prob +
           drive_alone + rural_cat + race_cat,
       data = ourbatch)
```

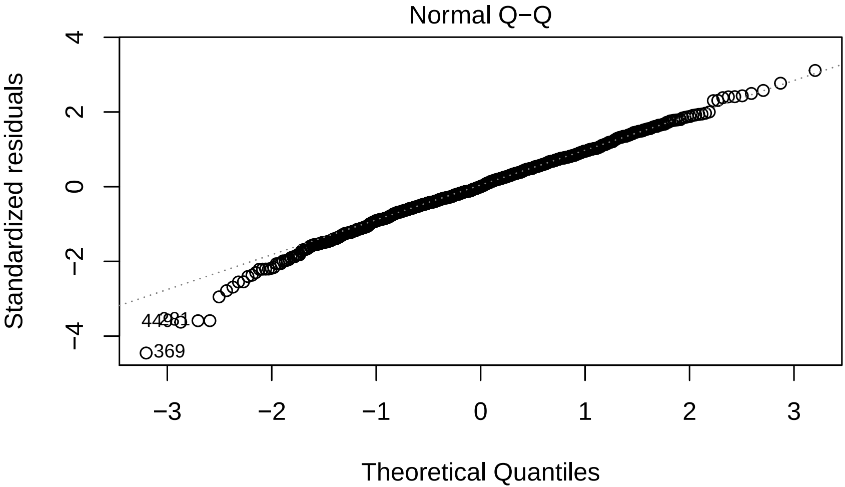
Quick Look at Residual Plots from `ks_log`?

```r
plot(ks_log, which = 1)
```

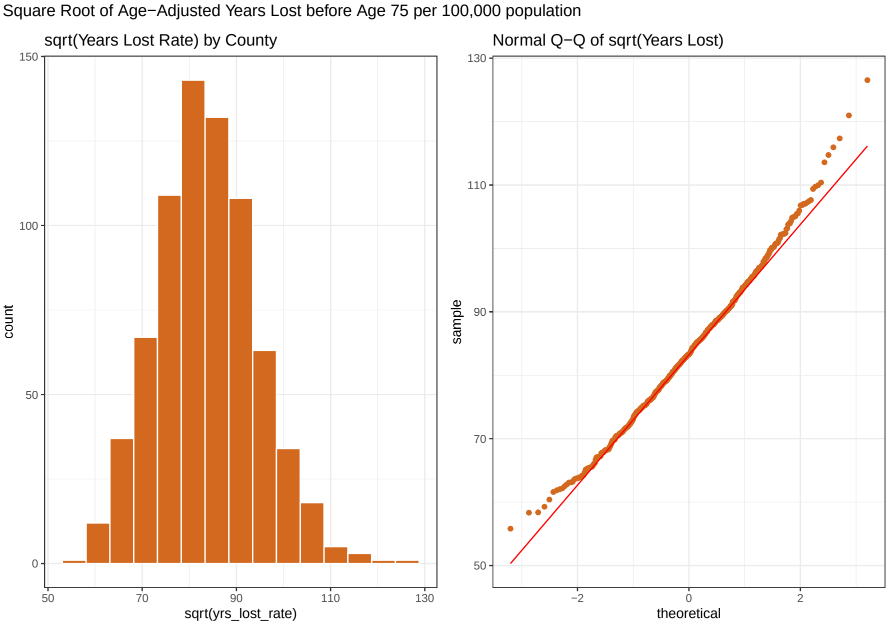


Quick Look at Residual Plots from `ks_log`?

```
plot(ks_log, which = 2)
```



How about a Square Root instead?

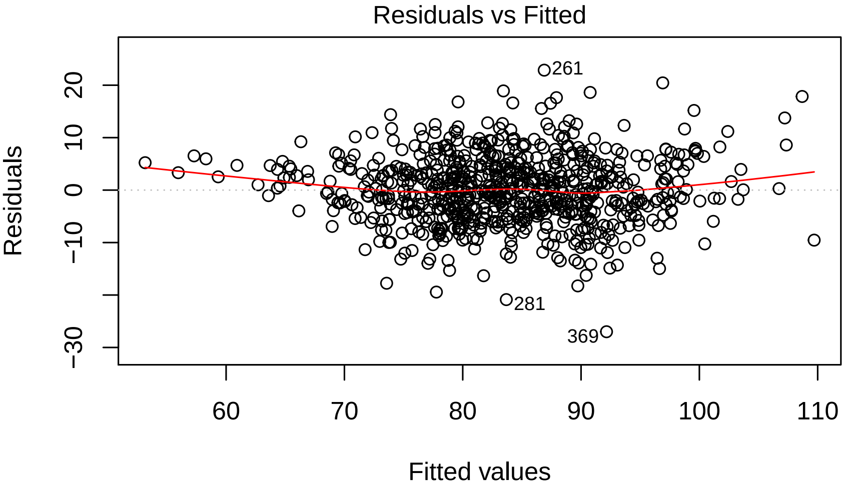


Kitchen Sink fit to `sqrt(yrs_lost_rate)`

```
ks_sqrt <-
    lm(sqrt(yrs_lost_rate) ~
           log_pop + not_english + hh_income +
           income_ratio + health_costs + social_assoc +
           pct_smokers + food_envir + housing_prob +
           drive_alone + rural_cat + race_cat,
       data = ourbatch)
```

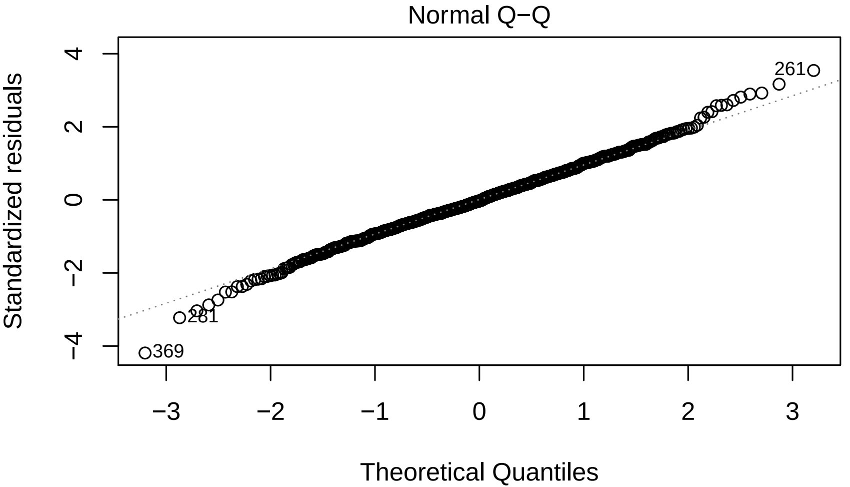
Quick Look at Residual Plots from `ks_sqrt`?

```
plot(ks_sqrt, which = 1)
```



Quick Look at Residual Plots from `ks_sqrt`?

```
plot(ks_sqrt, which = 2)
```



I'm going to go with the logarithm

```
ourbatch <- ourbatch %>%
  mutate(log_yrslost = log(yrs_lost_rate))

ourbatch %$% Hmisc::describe(log_yrslost)
```

```
log_yrslost
       n  missing distinct     Info     Mean      Gmd
     734        0      732        1     8.84   0.2849
     .05      .10      .25      .50      .75      .90
   8.413    8.511    8.669    8.845    9.003    9.150
     .95
   9.246

lowest : 8.044017 8.132354 8.133969 8.164425 8.201962
highest: 9.485150 9.506184 9.530248 9.591144 9.681044
```

Our "Model Selection" Problem

Let's focus for the moment on building a model to effectively predict our outcome `log_yrslost` using the 10 quantitative candidate predictors we've identified. (We'll tackle the categorical predictors separately, later.)

1. `log_pop`
2. `not_english`
3. `hh_income`
4. `income_ratio`
5. `health_costs`
6. `social_assoc`
7. `pct_smokers`
8. `food_envir`
9. `housing_prob`
10. `drive_alone`

We have 734 counties (observations) to use here.

Scatterplot Matrix?

We've got 10 quantitative predictors here, plus a (transformed) outcome.

Next, we'll show `ggpairs()` results (from `GGally` package)

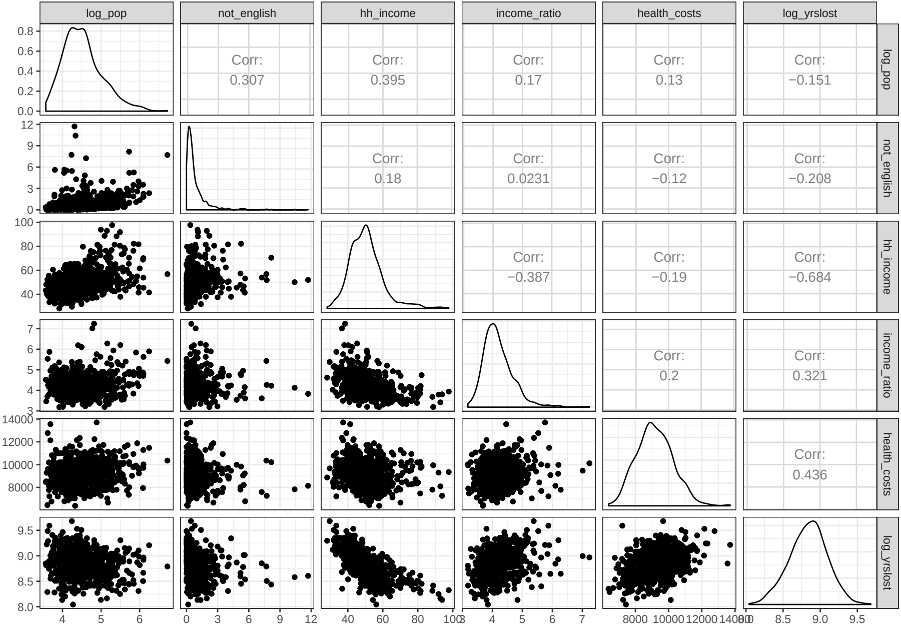
```
ggpairs(ourbatch, columns = c(17, 6:9, 18))
ggpairs(ourbatch, columns = c(10:14, 18))
```

- Run these with `message = FALSE`.
- Why include variable 18 twice, and at the end?
  - Variable order can be seen with `glimpse` or `names`.

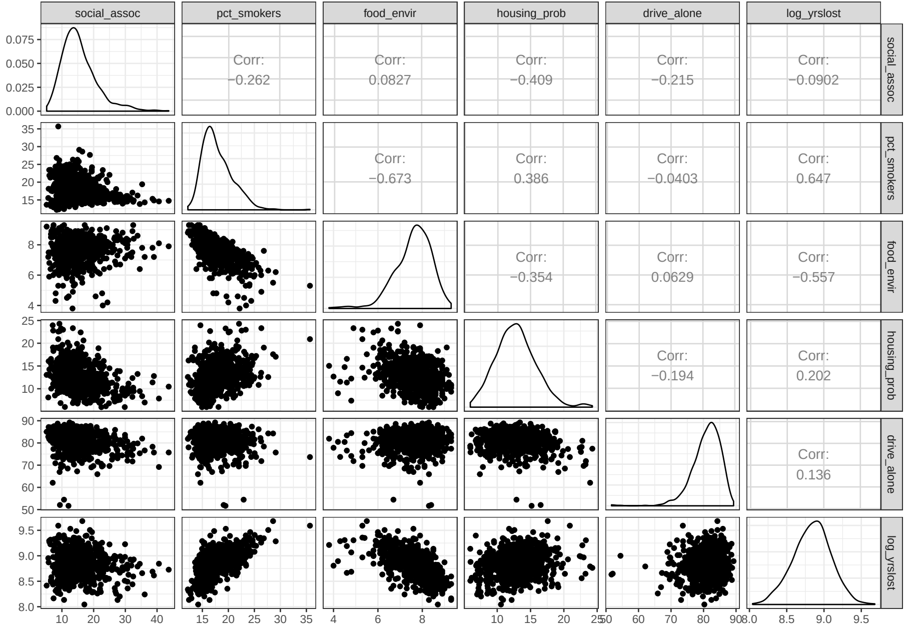
```
names(ourbatch)
```

```
 [1] "fips"          "state"         "county"
 [4] "yrs_lost_rate" "population"    "not_english"
 [7] "hh_income"     "income_ratio"  "health_costs"
[10] "social_assoc"  "pct_smokers"   "food_envir"
[13] "housing_prob"  "drive_alone"   "rural_cat"
[16] "race_cat"      "log_pop"       "log_yrslost"
```

Scatterplot Matrix with `GGally`, Part I



Scatterplot Matrix with GGally, Part II



Model Selection and “Best Subsets”

Stepwise Regression: Why It's a Bad Idea

- We have lots of candidate predictors.
- We don't have an especially large sample size.

It's tempting to apply "stepwise regression" to eliminate some of the predictors and produce a more parsimonious model. This would involve:

- deciding whether we would do forwards selection, backwards elimination or a combination
- using an R function called `step` to identify changes which minimize AIC at each step

For instance, we could start with a full model using all candidate predictors, and ask R to remove predictors, one at a time, whose removal most improved (reduced) the AIC.

- When R hit the point when removing another predictor would not improve AIC, it would stop and tell us what the "winning" model was.

So, what's the problem?

Stepwise Regression is a Bad Idea

Stepwise regression **encourages you not to think**.

- Most of the time, when you collect data, you have a reason to want to use it in your model.
- This algorithm's decisions about what variables to throw away / include doesn't actually minimize AIC over the set of possible models.

In addition, there are many more substantial problems (see Harrell 2001.) Models selected using stepwise regression look better than they are. . .

1. The R^2 values you get by doing this are biased high.
2. The parameter estimates are biased away from 0, their standard errors are too small, the confidence intervals around the estimates are too narrow, and the *p* values are too low.
3. Whatever collinearity issues you have in the data are made worse.

“All Subsets” or “Best Subsets”

A somewhat more plausible strategy is to consider all possible subsets of predictors, and search in a smart way for models which are good candidates because they optimize some combination of model fit summaries.

- The `leaps` package in R has a function called `regsubsets` to help us with this approach.

First, we specify a “full model” to `regsubsets` and list the maximum number of predictors we are willing to include in our models, with the `nvmax` parameter.

- `regsubsets` creates a subset selection object that identifies (by exhaustive search) the best models containing 1 predictor, then 2, then 3 and so forth up to `nvmax`.

Using regsubsets in leaps

- Here, we have 734 counties, and I'll consider models with up to 8 inputs.

```
rs_models <-
    regsubsets(log_yrslost ~ log_pop + not_english +
                         hh_income + income_ratio + health_costs +
                         social_assoc + pct_smokers + food_envir +
                         housing_prob + drive_alone,
                     data = ourbatch, nvmax = 8, nbest = 1)
```

- Other options available in regsubsets include
    - fitting more than one model per predictor count, via nbest
- The next slide displays rs_models.

```
Subset selection object
Call: regsubsets.formula(log_yrslost ~ log_pop + not_english +
    income_ratio + health_costs + social_assoc + pct_smokers +
    food_envir + housing_prob + drive_alone, data = ourbatch,
    nvmax = 8, nbest = 1)
10 Variables  (and intercept)
             Forced in Forced out
log_pop          FALSE      FALSE
not_english      FALSE      FALSE
hh_income        FALSE      FALSE
income_ratio     FALSE      FALSE
health_costs     FALSE      FALSE
social_assoc     FALSE      FALSE
pct_smokers      FALSE      FALSE
food_envir       FALSE      FALSE
housing_prob     FALSE      FALSE
drive_alone      FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
```

Summarizing Chosen Subsets

I've built a cleaner version of this on the next slide, by hand.

```
rs_summary <- summary(rs_models)
t(rs_summary$which)
```

```
                 1     2     3     4     5     6     7
(Intercept)   TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
log_pop      FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
not_english  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
hh_income     TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
income_ratio FALSE FALSE FALSE FALSE FALSE FALSE FALSE
health_costs FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
social_assoc FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
pct_smokers  FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
food_envir   FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
housing_prob FALSE FALSE FALSE FALSE FALSE FALSE FALSE
drive_alone  FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
                 8
(Intercept)   TRUE
```

Best Subsets from rs_summary

Model	Predictors
1	hh_income
2	hh_income, health_costs
3	Model 2 + pct_smokers
4	Model 3 + drive_alone
5	Model 4 + social_assoc
6	Model 5 + log_pop
7	Model 6 + food_envir
8	Model 7 + housing_prob

- income_ratio and not_english did not make any of these models
- As it turns out, in this situation, each model adds a predictor to the previous one, but that doesn't have to be the case.

Obtaining Fit Quality Statistics

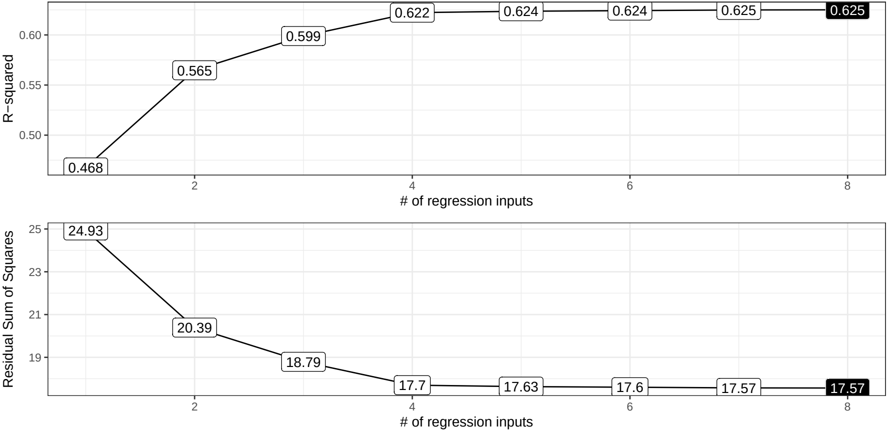
```r
rs_winners <- tbl_df(rs_summary$which) %>%
    mutate(inputs = 1:(rs_models$nvmax - 1),
           r2 = rs_summary$rsq,
           adjr2 = rs_summary$adjr2,
           cp = rs_summary$cp,
           bic = rs_summary$bic,
           rss = rs_summary$rss) %>%
    select(inputs, adjr2, cp, bic, everything())
```

Looking at rs_winners

```
rs_winners %>% slice(1:4)
```

```
# A tibble: 4 x 17
  inputs adjr2     cp   bic `(Intercept)` log_pop
   <int> <dbl>  <dbl> <dbl> <lgl>         <lgl>
1      1 0.467 297.   -450. TRUE          FALSE
2      2 0.564 112.   -591. TRUE          FALSE
3      3 0.597  48.1  -644. TRUE          FALSE
4      4 0.620   5.06 -681. TRUE          FALSE
# ... with 11 more variables: not_english <lgl>,
#   hh_income <lgl>, income_ratio <lgl>,
#   health_costs <lgl>, social_assoc <lgl>,
#   pct_smokers <lgl>, food_envir <lgl>,
#   housing_prob <lgl>, drive_alone <lgl>, r2 <dbl>,
#   rss <dbl>
```

Plotting Raw R^2 and Residual SS values



- These plots will always select the largest model available. Why?

So, which Subsets Look Best?

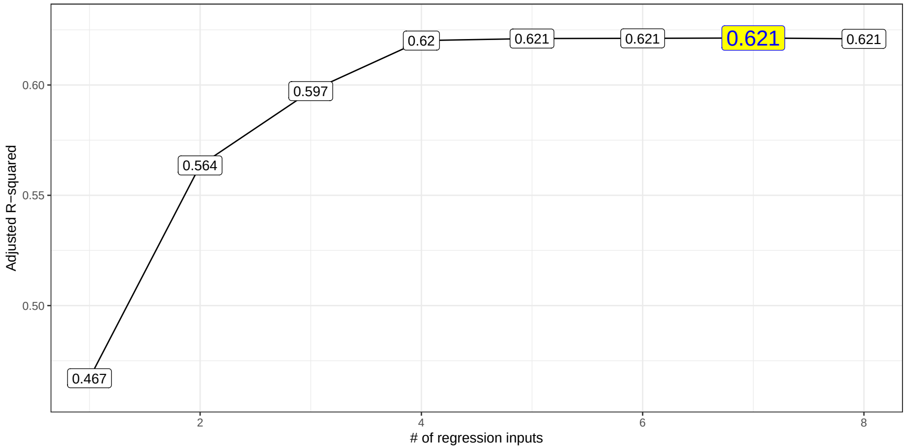
The `regsubsets` summary includes three model quality statistics:

- Adjusted R^2, which we'd like to maximize
- Mallows' C_p, which we'd like to minimize
- BIC, the Bayes Information Criterion, which we'd like to minimize

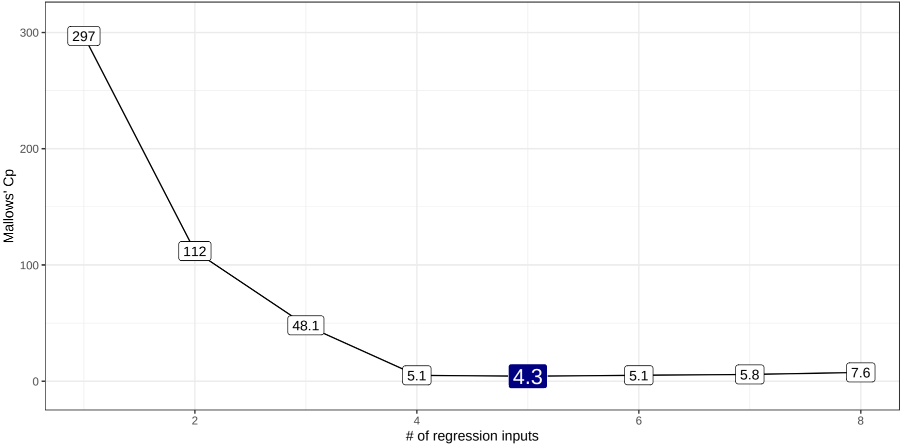
Plotting Adjusted R^2 values

```r
ggplot(rs_winners, aes(x = inputs, y = adjr2,
                       label = round(adjr2,3))) +
    geom_line() +
    geom_label() +
    geom_label(data = subset(rs_winners,
                             adjr2 == max(adjr2)),
               aes(x = inputs, y = adjr2,
                   label = round(adjr2,3)),
               fill = "yellow", col = "blue", size = 6) +
    scale_y_continuous(expand = expand_scale(mult = .1)) +
    labs(x = "# of regression inputs",
         y = "Adjusted R-squared")
```

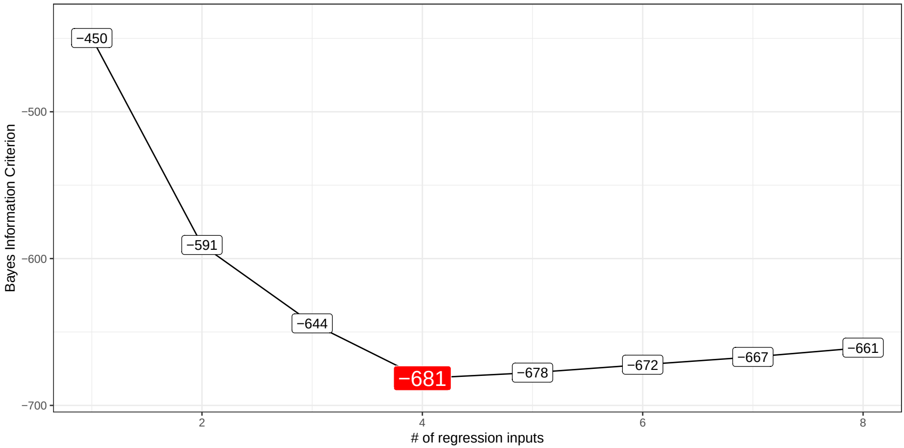
Adjusted R^2 values for our subsets



Mallows' C_p for our subsets



BIC for our subsets



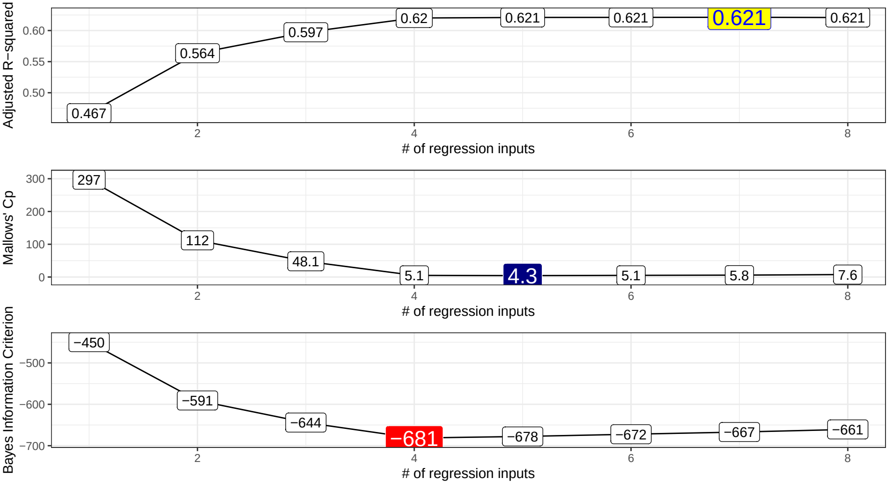
So, which Subsets Look Best?

We can use `which.max` and `which.min` to identify from the subset list the models selected by each of the three summaries we've been plotting…

```
tibble(AdjR2 = which.max(rs_summary$adjr2),
       Cp = which.min(rs_summary$cp),
       BIC = which.min(rs_summary$bic))
```

```
# A tibble: 1 x 3
  AdjR2    Cp   BIC
  <int> <int> <int>
1     7     5     4
```

By the plots, we have . . .



Coefficients for Candidate Models

```
coef(rs_models, id = 4)
```

```
  (Intercept)     hh_income  health_costs   pct_smokers
 7.899140e+00 -1.310119e-02  4.882626e-05  2.233060e-02
  drive_alone
 9.329765e-03
```

```
coef(rs_models, id = 5)
```

```
  (Intercept)     hh_income  health_costs  social_assoc
 8.034529e+00 -1.370597e-02  4.904949e-05 -1.990994e-03
  pct_smokers   drive_alone
 1.991591e-02  8.924561e-03
```

Coefficients for Candidate Models (continued)

```r
coef(rs_models, id = 7)
```

```
  (Intercept)        log_pop      hh_income  health_costs
 8.189596e+00 -1.962619e-02 -1.293658e-02  4.946928e-05
 social_assoc    pct_smokers     food_envir   drive_alone
-2.693840e-03  1.873528e-02 -1.188579e-02  9.089601e-03
```

Our Candidate Models

Inputs	Predictors
4	`hh_income`, `health_costs`, `pct_smokers`, `drive_alone`
5	Model 4 + `social_assoc`
6	Model 5 + `log_pop` and `food_envir`

In-Sample Comparisons of our Candidate Models

```r
m4 <- ourbatch %$%
    lm(log_yrslost ~ hh_income + health_costs +
           pct_smokers + drive_alone)

m5 <- ourbatch %$%
    lm(log_yrslost ~ hh_income + health_costs +
           pct_smokers + drive_alone + social_assoc)

m7 <- ourbatch %$%
    lm(log_yrslost ~ hh_income + health_costs +
           pct_smokers + drive_alone + social_assoc +
           log_pop + food_envir)
```

Models are **nested** so comparisons within samples are straightforward.

Comparing the in-sample fits of the models

```
bind_rows(glance(m4), glance(m5), glance(m7)) %>%
    mutate(model = c("m4", "m5", "m7")) %>%
    select(model, r2 = r.squared, adjr2 = adj.r.squared,
           AIC, BIC, sigma) %>%
    kable(digits = c(0, 3, 4, 1, 1, 4))
```

model	r2	adjr2	AIC	BIC	sigma
m4	0.622	0.6201	-639.2	-611.6	0.1558
m5	0.624	0.6211	-639.9	-607.8	0.1556
m7	0.625	0.6213	-638.5	-597.1	0.1556

Comparisons in-sample with anova

```r
anova(m4, m5, m7)
```

```
Analysis of Variance Table

Model 1: log_yrslost ~ hh_income + health_costs + pct_smokers
Model 2: log_yrslost ~ hh_income + health_costs + pct_smokers
    social_assoc
Model 3: log_yrslost ~ hh_income + health_costs + pct_smokers
    social_assoc + log_pop + food_envir
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1    729 17.699
2    728 17.632  1  0.067272 2.7794 0.09591 .
3    726 17.572  2  0.059984 1.2392 0.29024
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What if the models you're comparing aren't nested?

What if you're comparing:

- Model A: `lm(y = x1 + x2 + x3, data = dataset)`
- Model B: `lm(y = x1 + x4 + x5, data = dataset)`

Then . . .

- default *p* values from the ANOVA table comparing Model A to Model B aren't reasonable
- AIC and BIC are OK, can also used adjusted R^2 to help make a decision within the model building sample
- Still useful to think about out-of-sample prediction and cross-validation

Cross-Validation by Validation Split

A First Cross-Validation Approach

We'll start with a validation split. . .

1. Partition the data into a training set and a test set.
   - We'll identify a random sample of 80% of the `ourbatch` data into our training sample, and the rest will be in our test sample.
2. Build (train) the three candidate models we identified with "best subsets" by running them on the training set.
   - Note that I actually used the whole sample (not just the training sample) to perform best subsets here. I'm not claiming that was a good idea, just demonstrating.
3. Evaluate the model by looking at its effectiveness in the test set.
   - In this case, I'll compare the validation results for the three models we identified.

Measures of Fit Quality used in Cross-Validation

1. R-squared (R^2): the squared correlation between the observed outcome values and the predicted values by the model.
2. Root Mean Squared Error (RMSE), which measures the average prediction error made by the model in predicting the outcome for an observation. That is, the average difference between the observed known outcome values and the values predicted by the model. The lower the RMSE, the better the model.
3. Mean Absolute Error (MAE), an alternative to the RMSE that is less sensitive to outliers. It corresponds to the average absolute difference between observed and predicted outcomes. The lower the MAE, the better the model.

See the link on the Class 7 README for more details.

Validation Split Approach

We'll use the `createDataPartition` function from the `caret` package.

```r
set.seed(432)
training.samples <- ourbatch$log_yrslost %>%
    createDataPartition(p = 0.8, list = FALSE)

our_train <- ourbatch[training.samples,]
our_test <- ourbatch[-training.samples,]
```

Remember that our models are built to predict `log_yrs_lost` = `log(yrs_lost_rate)`, but we actually want to predict the `yrs_lost_rate`, so we'll need to do some back-transformation after making predictions.

Validation Split Approach for m4

1. Run the m4 model in the training sample.

```
m4_train <- lm(log_yrslost ~ hh_income + health_costs +
                   pct_smokers + drive_alone,
               data = our_train)
```

2. Obtain predicted (log(yrs_lost_rate)) values in the test sample.

```
m4_pred_logs <- m4_train %>% predict(our_test)
```

3. Exponentiate the predicted (log(yrs_lost_rate)) values to get predicted yrs_lost_rate values within the test sample.

```
m4_pred_yrslost <- exp(m4_pred_logs)
```

Validation Split Approach for m4 (continued)

4. Use the R2, RMSE and MAE functions from the caret package to calculate key summaries of the predictions we made for yrs_lost_rate using our model 4, and how they compare to the observed yrs_lost_rate values in the training sample.

```r
m4_summaries <- tibble(
    model = "m4",
    R2 = R2(m4_pred_yrslost, our_test$yrs_lost_rate),
    RMSE = RMSE(m4_pred_yrslost, our_test$yrs_lost_rate),
    MAE = MAE(m4_pred_yrslost, our_test$yrs_lost_rate),
    pred_err_rate = RMSE / mean(our_test$yrs_lost_rate)
)
```

Validation Split for Model m4: Results

```
m4_summaries
```

```
# A tibble: 1 x 5
  model    R2  RMSE   MAE pred_err_rate
  <chr> <dbl> <dbl> <dbl>         <dbl>
1 m4    0.456 1253.  997.         0.178
```

Validation Split for Model m5

```r
m5_train <- lm(log_yrslost ~ hh_income + health_costs +
                   pct_smokers + drive_alone + social_assoc,
               data = our_train)

m5_pred_logs <- m5_train %>% predict(our_test)
m5_pred_yrslost <- exp(m5_pred_logs)

m5_summaries <- tibble(
    model = "m5",
    R2 = R2(m5_pred_yrslost, our_test$yrs_lost_rate),
    RMSE = RMSE(m5_pred_yrslost, our_test$yrs_lost_rate),
    MAE = MAE(m5_pred_yrslost, our_test$yrs_lost_rate),
    pred_err_rate = RMSE / mean(our_test$yrs_lost_rate)
)
```

Validation Split for Model m7

```r
m7_train <- lm(log_yrslost ~ hh_income + health_costs +
                   pct_smokers + drive_alone +
                   social_assoc + log_pop + food_envir,
               data = our_train)

m7_pred_logs <- m7_train %>% predict(our_test)
m7_pred_yrslost <- exp(m7_pred_logs)

m7_summaries <- tibble(
    model = "m7",
    R2 = R2(m7_pred_yrslost, our_test$yrs_lost_rate),
    RMSE = RMSE(m7_pred_yrslost, our_test$yrs_lost_rate),
    MAE = MAE(m7_pred_yrslost, our_test$yrs_lost_rate),
    pred_err_rate = RMSE / mean(our_test$yrs_lost_rate)
)
```

Validation Split Approach Model Comparison

```
bind_rows(m4_summaries, m5_summaries, m7_summaries) %>%
    kable(digits = c(0, 3, 1, 1, 4))
```

model	R2	RMSE	MAE	pred_err_rate
m4	0.456	1252.9	996.6	0.1780
m5	0.457	1250.8	998.5	0.1777
m7	0.458	1250.3	996.9	0.1776

- Which of these models shows the strongest performance?

Validation Split Approach Model Comparison

```
bind_rows(m4_summaries, m5_summaries, m7_summaries) %>%
    kable(digits = c(0, 3, 1, 1, 4))
```

model	R2	RMSE	MAE	pred_err_rate
m4	0.456	1252.9	996.6	0.1780
m5	0.457	1250.8	998.5	0.1777
m7	0.458	1250.3	996.9	0.1776

- In validation R^2, we're looking at the squared correlation coefficient for the predictions and the observed values of `yrs_lost_rate`, so we want larger numbers.
    - Note that these validated R^2 values are not comparable to our earlier R^2 values, because now we're trying to predict `yrs_lost_rate` rather than `log(yrs_lost_rate)`.
    - Here, m7 is largest, followed by m5 and m4.

Validation Split Approach Model Comparison

```r
bind_rows(m4_summaries, m5_summaries, m7_summaries) %>%
    kable(digits = c(0, 3, 1, 1, 4))
```

model	R2	RMSE	MAE	pred_err_rate
m4	0.456	1252.9	996.6	0.1780
m5	0.457	1250.8	998.5	0.1777
m7	0.458	1250.3	996.9	0.1776

- The Root Mean Square Errors (RMSE) are similar, with m7 slightly smaller (better) then m5 and m4.
- The Mean Absolute Errors (MAE) in the validation sample are also similar, with m4 now a little smaller (better) than m7 or m5.
- The prediction error rate is just the RMSE divided by the mean of the actual `yrs_lost_rate`. No new information here on model fit.

K-Fold Cross Validation

Algorithm for K-fold Cross-Validation

1. Randomly split the data set into k subsets (for k-fold validation)
2. Reserve one subset and train the model on all other subsets
3. Test the model on the reserved subset and record the prediction error
4. Repeat this process until each of the k subsets has served as the test set.
5. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model.

Usually k is 5 or 10.

10-fold cross validation for model `m4`

```r
# Define training control
set.seed(43201)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model4_cv <- train(log_yrslost ~ hh_income + health_costs +
                   pct_smokers + drive_alone,
               data = ourbatch, method = "lm",
               trControl = train.control)
```

Summarize the results from `model4_cv`

```
model4_cv
```

```
Linear Regression 

734 samples
  4 predictor

No pre-processing
Resampling: Cross-Validated (10 fold) 
Summary of sample sizes: 661, 659, 660, 659, 661, 661, ... 
Resampling results:

  RMSE       Rsquared   MAE      
  0.1555325  0.6284756  0.1218445

Tuning parameter 'intercept' was held constant at a
 value of TRUE
```

10-fold cross validation for model m5

```r
# Define training control
set.seed(43202)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model5_cv <- train(log_yrslost ~ hh_income + health_costs +
                       pct_smokers + drive_alone +
                       social_assoc,
                   data = ourbatch, method = "lm",
                   trControl = train.control)
```

Summarize the results from `model5_cv`

```
model5_cv
```

```
Linear Regression 

734 samples
  5 predictor

No pre-processing
Resampling: Cross-Validated (10 fold) 
Summary of sample sizes: 660, 661, 660, 661, 660, 661, ... 
Resampling results:

  RMSE       Rsquared   MAE      
  0.1557501  0.6272377  0.1222321

Tuning parameter 'intercept' was held constant at a
 value of TRUE
```

10-fold cross validation for model `m7`

```r
# Define training control
set.seed(43203)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model7_cv <- train(log_yrslost ~ hh_income + health_costs +
                       pct_smokers + drive_alone +
                       social_assoc + log_pop + food_envir,
                   data = ourbatch, method = "lm",
                   trControl = train.control)
```

Summarize the results from `model7_cv`

```
model7_cv
```

```
Linear Regression

734 samples
  7 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 660, 658, 662, 659, 662, 660, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.1567999  0.6207551  0.122406

Tuning parameter 'intercept' was held constant at a
 value of TRUE
```

Comparing 10-fold Cross-Validation Results

```
res_cv10 <- bind_rows(
    model4_cv$results,
    model5_cv$results,
    model7_cv$results) %>%
    mutate(model = c("m4", "m5", "m7")) %>%
    select(model, RMSE, Rsquared, MAE)

res_cv10 %>% kable(digits = c(0, 4, 3, 4))
```

model	RMSE	Rsquared	MAE
m4	0.1555	0.628	0.1218
m5	0.1558	0.627	0.1222
m7	0.1568	0.621	0.1224

Still to Come

1. What happens when you include categorical variables into best subsets?
2. The Spearman ρ^2 plot as a way to identify potential variables to consider non-linear terms.
3. Regression Diagnostics for Linear Models
4. What if we want to predict a binary outcome? Can we use `bestglm`?
5. What if we want to include sampling weights?
6. What about multiple imputation?