Case study: asphalt

The asphalt data

- 31 asphalt pavements prepared under different conditions. How does quality of pavement depend on these?
- Variables:
  - `pct.a.surf` Percentage of asphalt in surface layer
  - `pct.a.base` Percentage of asphalt in base layer
  - `fines` Percentage of fines in surface layer
  - `voids` Percentage of voids in surface layer
  - `rut.depth` Change in rut depth per million vehicle passes
  - `viscosity` Viscosity of asphalt
  - `run` 2 data collection periods: 1 for run 1, 0 for run 2.
- `rut.depth` response. Depends on other variables, how?

Packages for this section

```
library(MASS)
library(tidyverse)
library(broom)
library(leaps)
```

Make sure to load `MASS` before `tidyverse` (for annoying technical reasons).

Getting set up

```
my_url <- "http://ritsokiguess.site/datafiles/asphalt.txt"
asphalt <- read_delim(my_url, " ")
```

- Quantitative variables with one response: multiple regression.
- Some issues here that don't come up in "simple" regression; handle as we go. (STAB27/STAC67 ideas.)

The data (some)

```
asphalt
```

```
# A tibble: 31 x 7
   pct.a.surf pct.a.base fines voids rut.depth viscosity   run
        <dbl>      <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>
 1       4.68       4.87   8.4  4.92      6.75      2.8      1
 2       5.19       4.5    6.5  4.56     13         1.4      1
 3       4.82       4.73   7.9  5.32     14.8       1.4      1
 4       4.85       4.76   8.3  4.86     12.6       3.3      1
 5       4.86       4.95   8.4  3.78      8.25      1.7      1
 6       5.16       4.45   7.4  4.40     10.7       2.9      1
 7       4.82       5.05   6.8  4.87      7.28      3.7      1
 8       4.86       4.7    8.6  4.83     12.7       1.7      1
 9       4.78       4.84   6.7  4.86     12.6       0.92     1
10       5.16       4.76   7.7  4.03     20.6       0.68     1
# i 21 more rows
```

Plotting response "rut depth" against everything else

Same idea as for plotting separate predictions on one plot:

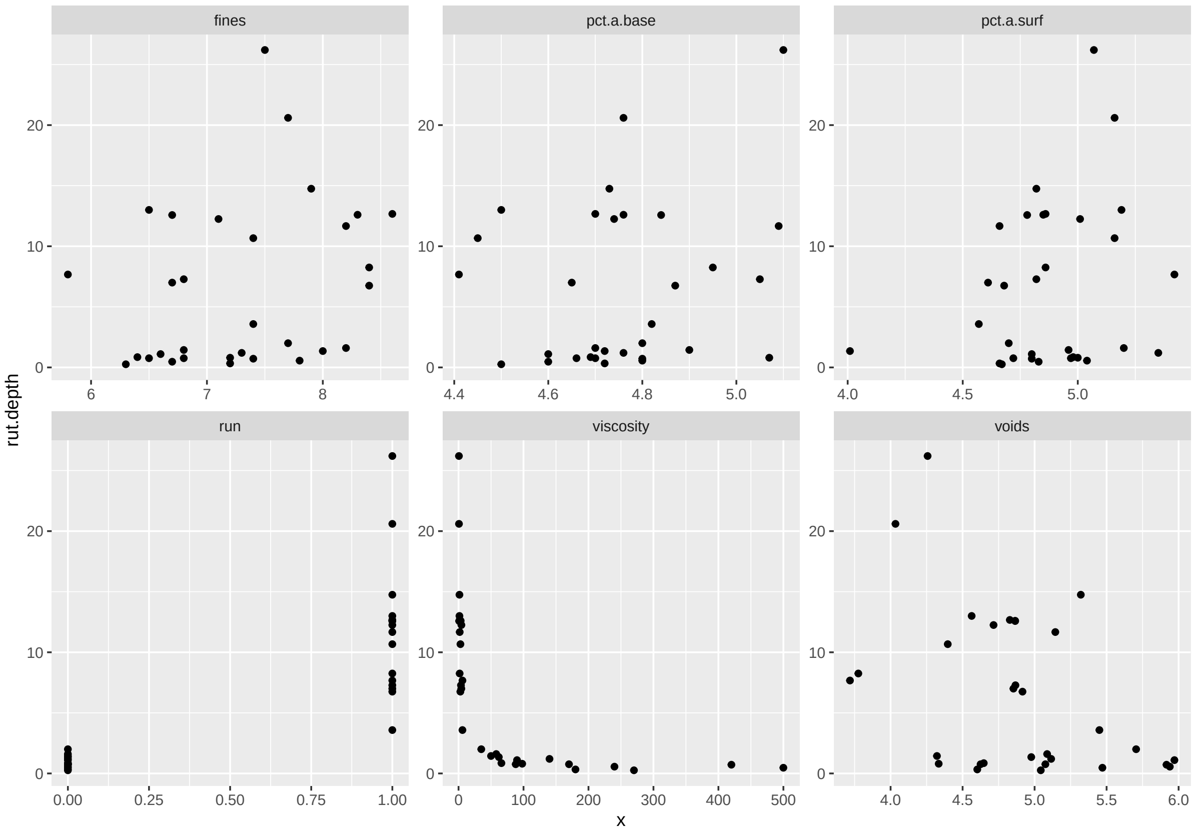
```
asphalt %>%
  pivot_longer(
    -rut.depth,
    names_to="xname", values_to="x"
  ) %>%
  ggplot(aes(x = x, y = rut.depth)) + geom_point() +
  facet_wrap(~xname, scales = "free") -> g
```

"collect all the x-variables together into one column called x, with another column xname saying which x they were, then plot these x's against rut.depth, a separate facet for each x-variable."

I saved this graph to plot later (on the next page).

The plot

```
g
```



Interpreting the plots

- One plot of rut depth against each of the six other variables.
- Get rough idea of what's going on.
- Trends mostly weak.
- `viscosity` has strong but non-linear trend.
- `run` has effect but variability bigger when run is 1.
- Weak but downward trend for `voids`.
- Non-linearity of `rut.depth`-`viscosity` relationship should concern us.

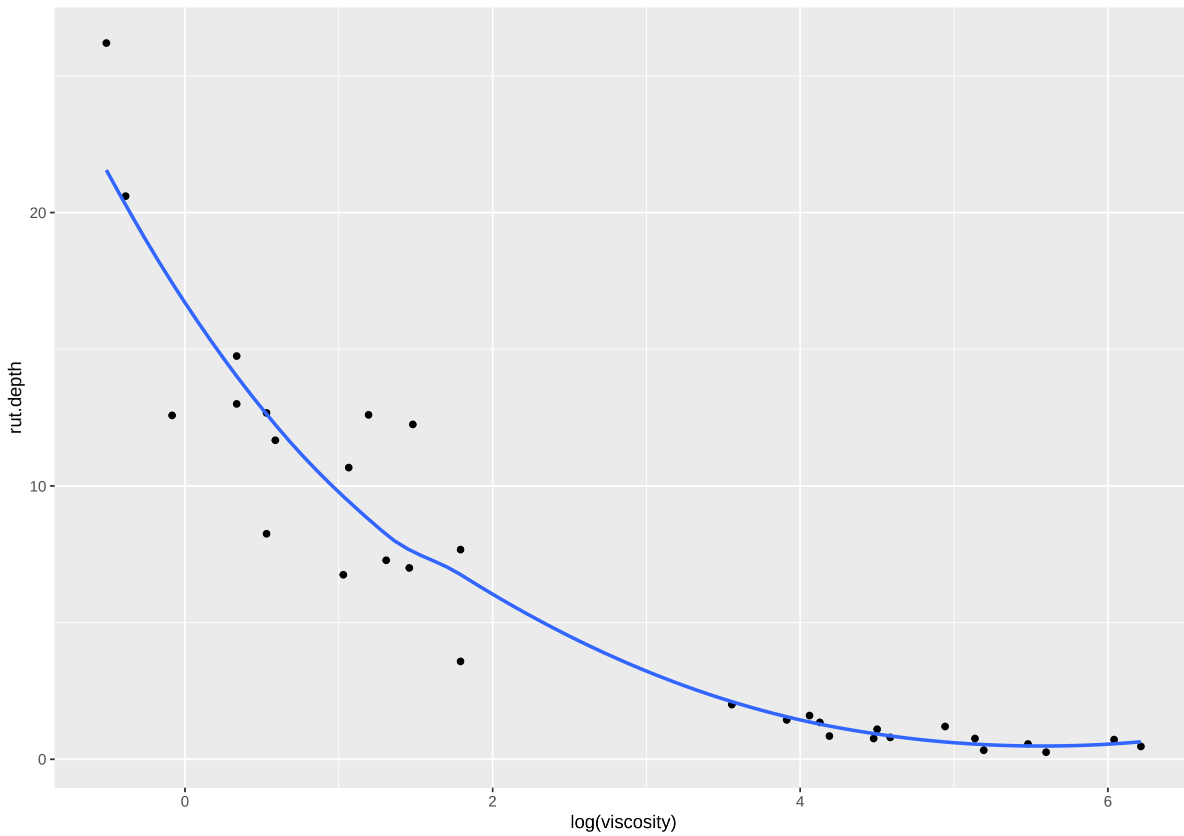
Log of `viscosity`: more nearly linear?

- Take this back to asphalt engineer: suggests log of `viscosity`:

```
ggplot(asphalt, aes(y = rut.depth, x = log(viscosity))) +
  geom_point() + geom_smooth(se = F)
```

(plot overleaf)

Rut depth against log-viscosity



Comments and next steps

- Not very linear, but better than before.
- In multiple regression, hard to guess which x's affect response. So typically start by predicting from everything else.
- Model formula has response on left, squiggle, explanatories on right joined by plusses:

```
rut.1 <- lm(rut.depth ~ pct.a.surf + pct.a.base + fines +
  voids + log(viscosity) + run, data = asphalt)
```

Regression output: `summary(rut.1)` or:

```
glance(rut.1)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic     p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>       <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.806         0.758  3.32      16.6 0.000000174     6  -77.3  171.  182.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
tidy(rut.1)
```

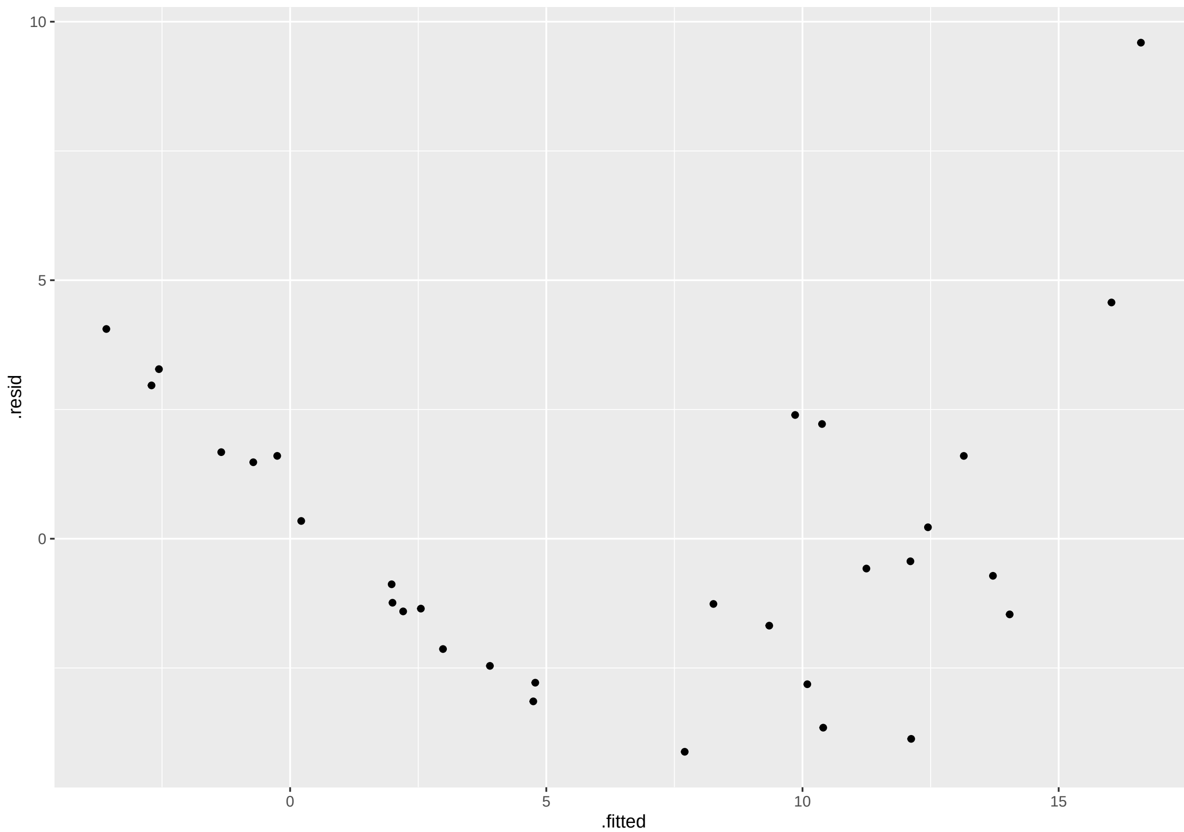
```
# A tibble: 7 x 5
  term           estimate std.error statistic p.value
  <chr>             <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)     -13.0      26.2      -0.496 0.625
2 pct.a.surf        3.97      2.50      1.59  0.125
3 pct.a.base        1.26      3.97      0.318 0.753
4 fines             0.116     1.01      0.115 0.909
5 voids             0.589     1.32      0.445 0.660
6 log(viscosity)   -3.15      0.919    -3.43  0.00220
7 run              -1.97      3.65     -0.539 0.595
```

Comments

- R-squared 81%, not so bad.
- P-value in `glance` asserts that something helping to predict rut.depth.
- Table of coefficients says `log(viscosity)`.
- But confused by clearly non-significant variables: remove those to get clearer picture of what is helpful.
- Before we do anything, look at residual plots:
  - (a) of residuals against fitted values (as usual)
  - (b) of residuals against each explanatory.
- Problem fixes:
  - with (a): fix response variable;
  - with some plots in (b): fix those explanatory variables.

Plot fitted values against residuals

```
ggplot(rut.1, aes(x = .fitted, y = .resid)) + geom_point()
```



Plotting residuals against x variables

- Problem here is that residuals are in the fitted model, and the observed x-values are in the original data frame `asphalt`.
- Package broom contains a function `augment` that combines these two together so that they can later be plotted: start with a model first, and then augment with a data frame:

```
rut.1 %>% augment(asphalt) -> rut.1a
```

What does rut.1a contain?

```
names(rut.1a)
```

```
 [1] "pct.a.surf" "pct.a.base" "fines"      "voids"      "rut.depth"
 [6] "viscosity"  "run"        ".fitted"    ".resid"     ".hat"
[11] ".sigma"     ".cooksd"    ".std.resid"
```

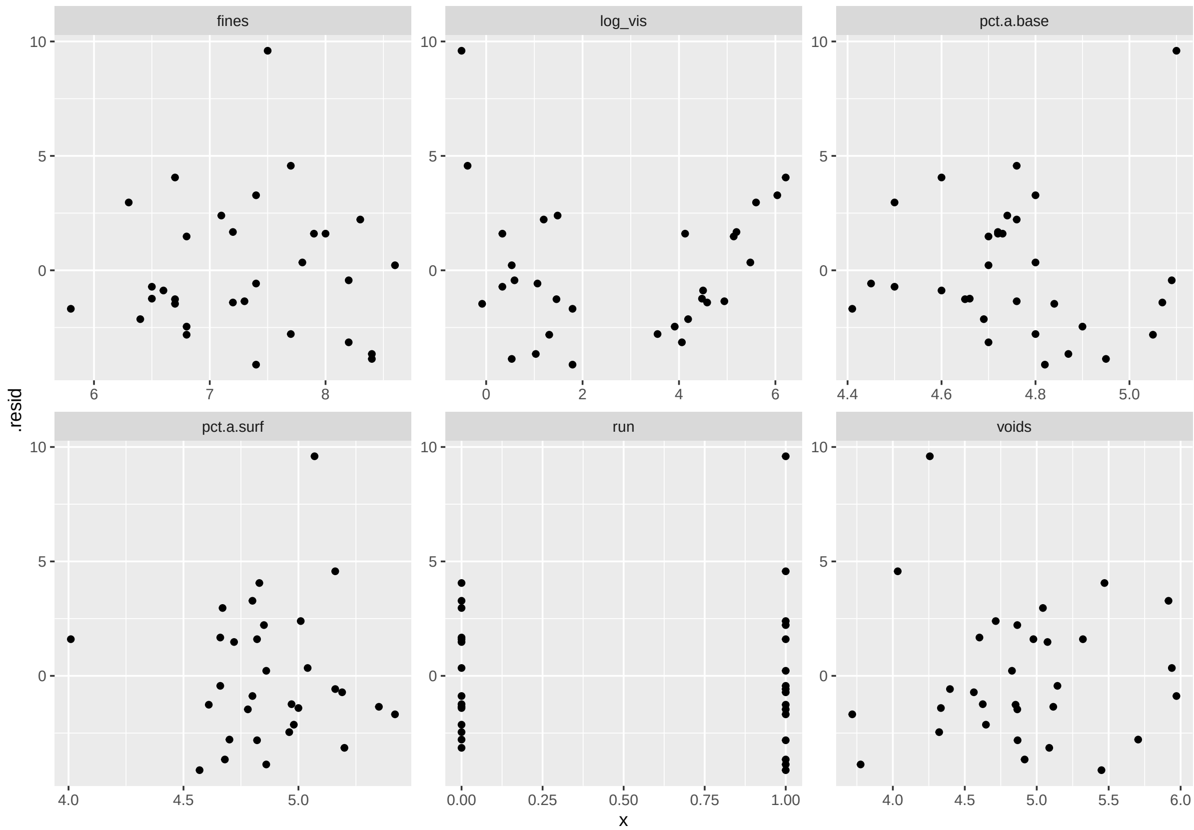
- all the stuff in original data frame, plus:
- quantities from regression (starting with a dot)

Plotting residuals against x-variables

```
rut.1a %>%
  mutate(log_vis=log(viscosity)) %>%
  pivot_longer(
    c(pct.a.surf:voids, run, log_vis),
    names_to="xname", values_to="x"
  ) %>%
  ggplot(aes(x = x, y = .resid)) +
  geom_point() + facet_wrap(~xname, scales = "free") -> g
```

The plot

```
g
```



Comments

- There is serious curve in plot of residuals vs. fitted values. Suggests a transformation of y.
- The residuals-vs-x's plots don't show any serious trends. Worst probably that potential curve against log-viscosity.
- Also, large positive residual, 10, that shows up on all plots. Perhaps transformation of y will help with this too.
- If residual-fitted plot OK, but some residual-x plots not, try transforming those x's, eg. by adding x^2 to help with curve.

Which transformation?

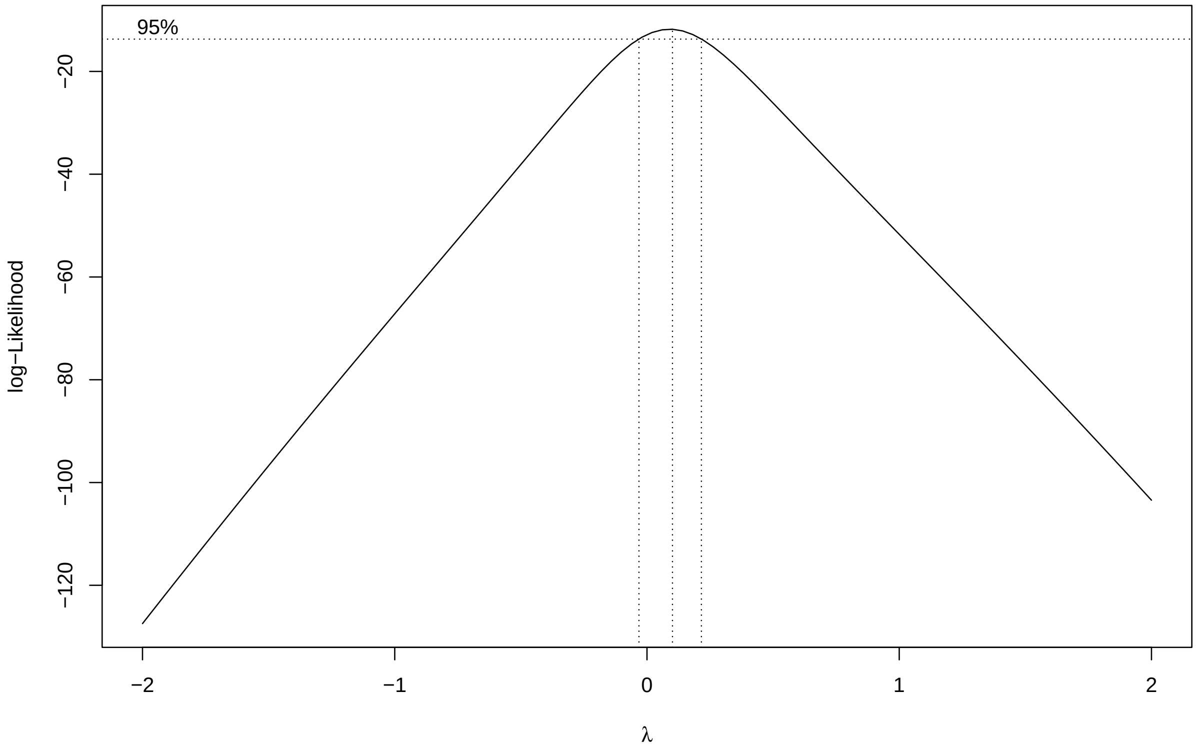
- Best way: consult with person who brought you the data.
- Can't do that here!
- No idea what transformation would be good.
- Let data choose: "Box-Cox transformation".

- Scale is that of "ladder of powers": power transformation, but 0 is log.

Running Box-Cox

From package `MASS`:

```
boxcox(rut.depth ~ pct.a.surf + pct.a.base + fines + voids +
  log(viscosity) + run, data = asphalt)
```



Comments on Box-Cox plot

- λ represents power to transform y with.
- Best single choice of transformation parameter λ is peak of curve, close to 0.
- Vertical dotted lines give CI for λ, about $(-0.05, 0.2)$.
- $\lambda = 0$ means "log".
- Narrowness of confidence interval mean that these not supported by data:
  - No transformation ($\lambda = 1$)
  - Square root ($\lambda = 0.5$)
  - Reciprocal ($\lambda = -1$).

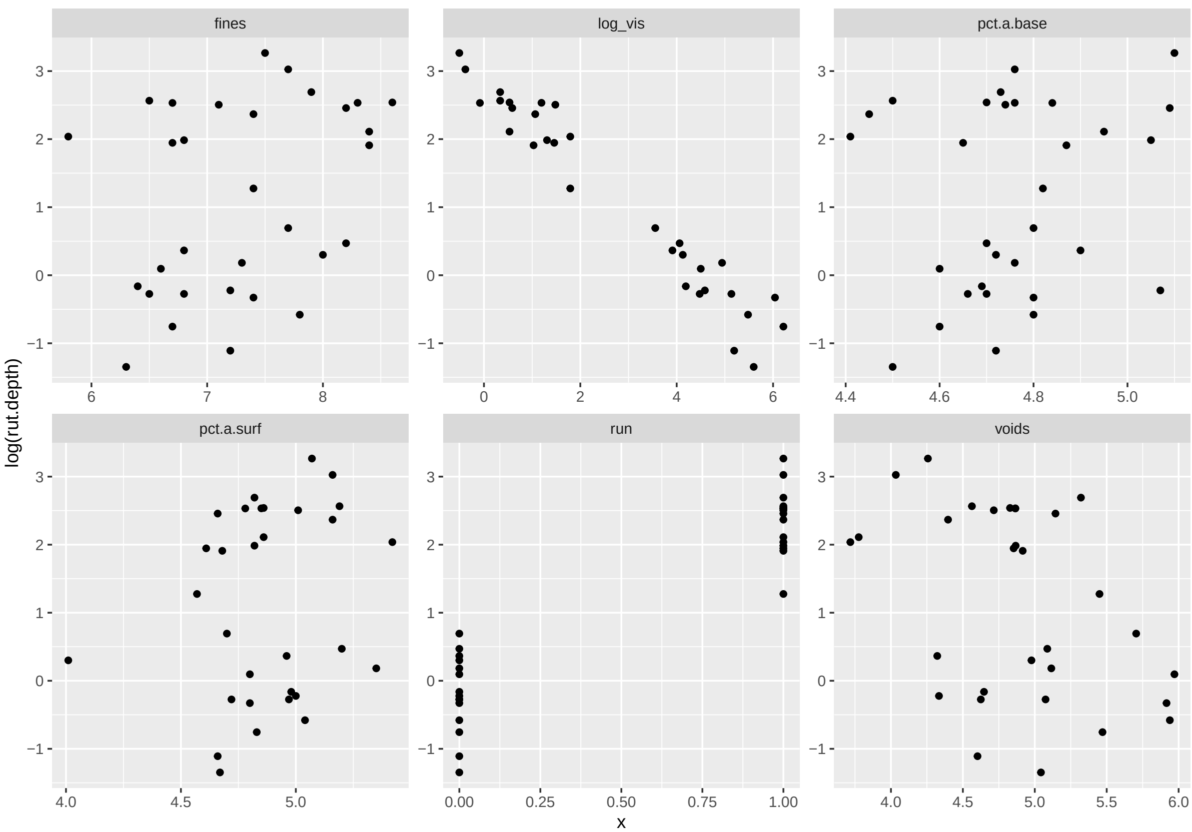
Relationships with explanatories

- As before: plot response (now `log(rut.depth)`) against other explanatory variables, all in one shot:

```
asphalt %>%
  mutate(log_vis=log(viscosity)) %>%
  pivot_longer(
    c(pct.a.surf:voids, run, log_vis),
    names_to="xname", values_to="x"
  ) %>%
  ggplot(aes(y = log(rut.depth), x = x)) + geom_point() +
  facet_wrap(~xname, scales = "free") -> g3
```

The new plots

```
g3
```



Modelling with transformed response

- These trends look pretty straight, especially with `log.viscosity`.
- Values of `log.rut.depth` for each `run` have same spread.
- Other trends weak, but are straight if they exist.
- Start modelling from the beginning again.
- Model `log.rut.depth` in terms of everything else, see what can be removed:

```
rut.2 <- lm(log(rut.depth) ~ pct.a.surf + pct.a.base +
  fines + voids + log(viscosity) + run, data = asphalt)
```

- use `tidy` from `broom` to display just the coefficients.

Output

```
tidy(rut.2)
```

```
# A tibble: 7 x 5
  term           estimate std.error statistic     p.value
  <chr>             <dbl>     <dbl>     <dbl>       <dbl>
1 (Intercept)     -1.57      2.44      -0.646 0.525
2 pct.a.surf       0.584     0.232      2.52  0.0190
3 pct.a.base      -0.103     0.369     -0.280 0.782
4 fines            0.0978    0.0941     1.04  0.309
5 voids            0.199     0.123      1.62  0.119
6 log(viscosity)  -0.558     0.0854    -6.53  0.000000945
7 run              0.340     0.339      1.00  0.326
```

Taking out everything non-significant

- Try: remove everything but pct.a.surf and log.viscosity:

```
rut.3 <- lm(log(rut.depth) ~ pct.a.surf + log(viscosity), data = asphalt)
```

- Check that removing all those variables wasn't too much:

```
anova(rut.3, rut.2)
```

```
Analysis of Variance Table

Model 1: log(rut.depth) ~ pct.a.surf + log(viscosity)
Model 2: log(rut.depth) ~ pct.a.surf + pct.a.base + fines + voids + log(viscosity) +
    run
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     28 2.8809
2     24 2.2888  4   0.59216 1.5523 0.2191
```

- H_0 : two models equally good; H_a : bigger model better.
- Null not rejected here; small model as good as the big one, so prefer simpler smaller model `rut.3`.

Find the largest P-value by eye:

```r
tidy(rut.2)
```

```
# A tibble: 7 x 5
  term           estimate std.error statistic     p.value
  <chr>             <dbl>     <dbl>     <dbl>       <dbl>
1 (Intercept)     -1.57      2.44      -0.646 0.525
2 pct.a.surf       0.584     0.232      2.52  0.0190
3 pct.a.base      -0.103     0.369     -0.280 0.782
4 fines            0.0978    0.0941     1.04  0.309
5 voids            0.199     0.123      1.62  0.119
6 log(viscosity)  -0.558     0.0854    -6.53  0.000000945
7 run              0.340     0.339      1.00  0.326
```

- Largest P-value is 0.78 for `pct.a.base`, not significant.
- So remove this first, re-fit and re-assess.
- Or, as over.

Get the computer to find the largest P-value for you

- Output from `tidy` is itself a data frame, thus:

```r
tidy(rut.2) %>% arrange(p.value)
```

```
# A tibble: 7 x 5
  term           estimate std.error statistic     p.value
  <chr>             <dbl>     <dbl>     <dbl>       <dbl>
```

```
1 log(viscosity)  -0.558     0.0854    -6.53  0.000000945
2 pct.a.surf       0.584     0.232      2.52  0.0190
3 voids            0.199     0.123      1.62  0.119
4 fines            0.0978    0.0941     1.04  0.309
5 run              0.340     0.339      1.00  0.326
6 (Intercept)     -1.57      2.44      -0.646 0.525
7 pct.a.base      -0.103     0.369     -0.280 0.782
```

- Largest P-value at the bottom.

Take out `pct.a.base`

- Copy and paste the `lm` code and remove what you're removing:

```r
rut.4 <- lm(log(rut.depth) ~ pct.a.surf + fines + voids +
              log(viscosity) + run, data = asphalt)
tidy(rut.4) %>% arrange(p.value)
```

```
# A tibble: 6 x 5
  term           estimate std.error statistic     p.value
  <chr>             <dbl>     <dbl>     <dbl>       <dbl>
1 log(viscosity)  -0.552     0.0818     -6.75 0.000000448
2 pct.a.surf       0.593     0.225       2.63 0.0143
3 voids            0.200     0.121       1.66 0.109
4 (Intercept)     -2.08      1.61       -1.29 0.208
5 run              0.360     0.325       1.11 0.279
6 fines            0.0889    0.0870      1.02 0.316
```

- `fines` is next to go, P-value 0.32.

"Update"

Another way to do the same thing:

```r
rut.4 <- update(rut.2, . ~ . - pct.a.base)
tidy(rut.4) %>% arrange(p.value)
```

```
# A tibble: 6 x 5
  term           estimate std.error statistic     p.value
  <chr>             <dbl>     <dbl>     <dbl>       <dbl>
1 log(viscosity)  -0.552     0.0818     -6.75 0.000000448
```

```
2 pct.a.surf        0.593     0.225       2.63 0.0143
3 voids             0.200     0.121       1.66 0.109
4 (Intercept)      -2.08      1.61       -1.29 0.208
5 run               0.360     0.325       1.11 0.279
6 fines             0.0889    0.0870      1.02 0.316
```

- Again, `fines` is the one to go. (Output identical as it should be.)

Take out fines:

```r
rut.5 <- update(rut.4, . ~ . - fines)
tidy(rut.5) %>% arrange(p.value)
```

```
# A tibble: 5 x 5
  term           estimate std.error statistic      p.value
  <chr>             <dbl>     <dbl>     <dbl>        <dbl>
1 log(viscosity)   -0.580    0.0772    -7.52  0.0000000559
2 pct.a.surf        0.548    0.221      2.48  0.0200
3 voids             0.232    0.117      1.99  0.0577
4 run               0.295    0.319      0.923 0.365
5 (Intercept)      -1.26     1.39      -0.902 0.375
```

Can't take out intercept, so `run`, with P-value 0.36, goes next.

Take out run:

```r
rut.6 <- update(rut.5, . ~ . - run)
tidy(rut.6) %>% arrange(p.value)
```

```
# A tibble: 4 x 5
  term           estimate std.error statistic  p.value
  <chr>             <dbl>     <dbl>     <dbl>    <dbl>
1 log(viscosity)   -0.646    0.0288   -22.5   5.29e-19
2 pct.a.surf        0.555    0.220      2.52  1.80e- 2
3 voids             0.245    0.116      2.12  4.36e- 2
4 (Intercept)      -1.02     1.36      -0.748 4.61e- 1
```

Again, can't take out intercept, so largest P-value is for `voids`, 0.044. But this is significant, so we shouldn't remove `voids`.

Comments

- Here we stop: `pct.a.surf`, `voids` and `log.viscosity` would all make fit significantly worse if removed. So they stay.
- Different final result from taking things out one at a time (top), than by taking out 4 at once (bottom):

```
coef(rut.6)
```

```
 (Intercept)     pct.a.surf          voids log(viscosity)
  -1.0207945      0.5554686      0.2447934     -0.6464911
```

```
coef(rut.3)
```

```
 (Intercept)     pct.a.surf log(viscosity)
   0.9001389      0.3911481     -0.6185628
```

- Point: Can make difference which way we go.

Comments on variable selection

- Best way to decide which x's belong: expert knowledge: which of them should be important.
- Best automatic method: what we did, "backward selection".
- Do not learn about "stepwise regression"! **eg. here**
- R has function `step` that does backward selection, like this:

```
step(rut.2, direction = "backward", test = "F")
```

Gets same answer as we did (by removing least significant x).

- Removing non-significant x's may remove interesting ones whose P-values happened not to reach 0.05. Consider using less stringent cutoff like 0.20 or even bigger.
- Can also fit all possible regressions, as over (may need to do `install.packages("leaps")` first).

All possible regressions (output over)

Uses package `leaps`:

```r
leaps <- regsubsets(log(rut.depth) ~ pct.a.surf +
                      pct.a.base + fines + voids +
                      log(viscosity) + run,
                    data = asphalt, nbest = 2)
s <- summary(leaps)
with(s, data.frame(rsq, outmat)) -> d
```

The output

```r
d %>% rownames_to_column("model") %>% arrange(desc(rsq))
```

```
      model       rsq pct.a.surf pct.a.base fines voids log.viscosity. run
1  6  ( 1 ) 0.9609642          *          *     *     *              *   *
2  5  ( 1 ) 0.9608365          *                *     *              *   *
3  5  ( 2 ) 0.9593265          *          *     *     *              *
4  4  ( 1 ) 0.9591996          *                      *              *   *
5  4  ( 2 ) 0.9589206          *                *     *              *
6  3  ( 1 ) 0.9578631          *                      *              *
7  3  ( 2 ) 0.9534561          *                *                    *
8  2  ( 1 ) 0.9508647          *                                     *
9  2  ( 2 ) 0.9479541                                 *              *
10 1  ( 1 ) 0.9452562                                                *
11 1  ( 2 ) 0.8624107                                                    *
```

Comments

- Problem: even adding a worthless x increases R-squared. So try for line where R-squared stops increasing "too much", eg. top line (just log.viscosity), first 3-variable line (backwards-elimination model). Hard to judge.
- One solution (STAC67): adjusted R-squared, where adding worthless variable makes it go down.
- `data.frame` rather than `tibble` because there are several columns in `outmat`.

All possible regressions, adjusted R-squared

```r
with(s, data.frame(adjr2, outmat)) %>%
  rownames_to_column("model") %>%
  arrange(desc(adjr2))
```

```
      model     adjr2 pct.a.surf pct.a.base fines voids log.viscosity. run
1  3  ( 1 ) 0.9531812          *                      *              *
2  5  ( 1 ) 0.9530038          *                *     *              *   *
3  4  ( 1 ) 0.9529226          *                      *              *   *
```

```
4  4  ( 2 ) 0.9526007         *           *    *           *
5  6  ( 1 ) 0.9512052         *      *    *    *           *  *
6  5  ( 2 ) 0.9511918         *      *    *    *           *
7  3  ( 2 ) 0.9482845         *           *                *
8  2  ( 1 ) 0.9473550         *                            *
9  2  ( 2 ) 0.9442365                      *                *
10 1  ( 1 ) 0.9433685                                       *
11 1  ( 2 ) 0.8576662                                          *
```

Revisiting the best model

- Best model was our rut.6:

```
tidy(rut.6)
```

```
# A tibble: 4 x 5
  term           estimate std.error statistic  p.value
  <chr>             <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)      -1.02     1.36      -0.748 4.61e- 1
2 pct.a.surf        0.555    0.220      2.52  1.80e- 2
3 voids             0.245    0.116      2.12  4.36e- 2
4 log(viscosity)   -0.646    0.0288   -22.5   5.29e-19
```

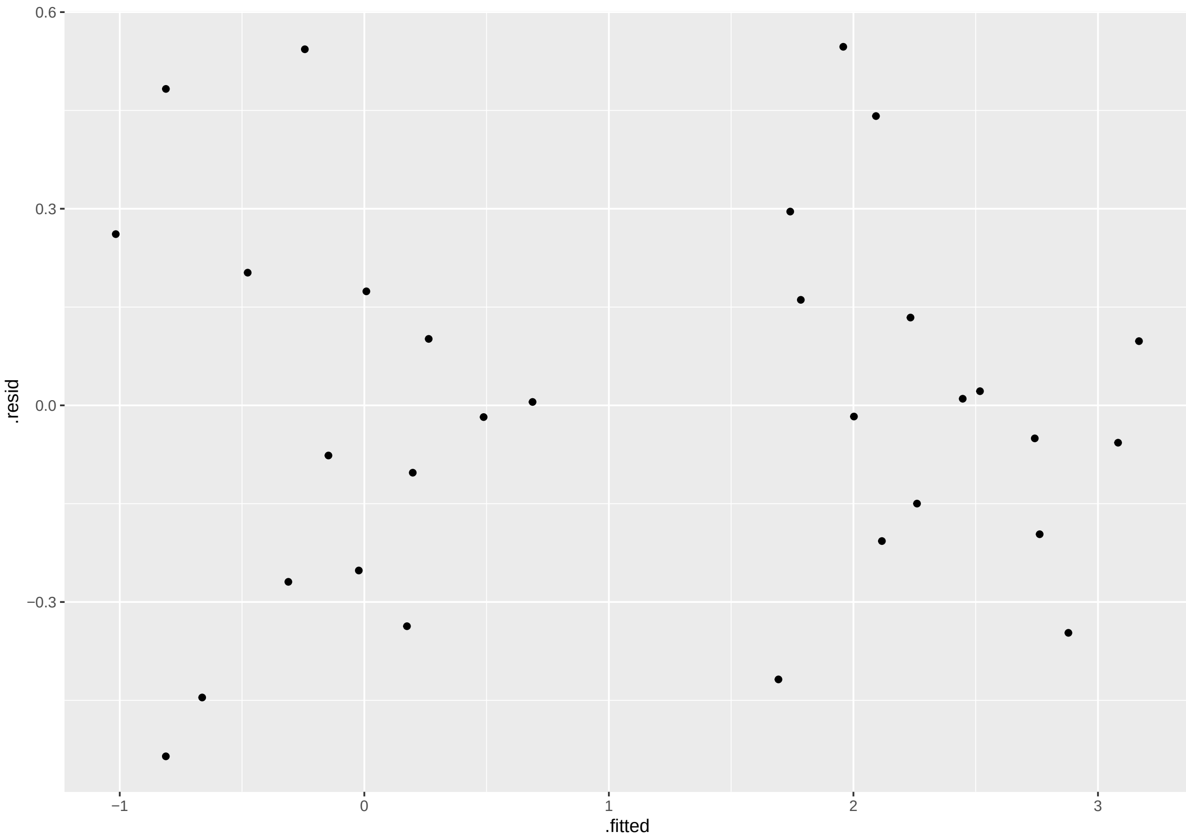
Revisiting (2)

- Regression slopes say that rut depth increases as log-viscosity decreases, `pct.a.surf` increases and `voids` increases. This more or less checks out with out scatterplots against `log.viscosity`.
- We should check residual plots again, though previous scatterplots say it's unlikely that there will be a problem:

```
g <- ggplot(rut.6, aes(y = .resid, x = .fitted)) +
geom_point()
```

Residuals against fitted values

```
g
```



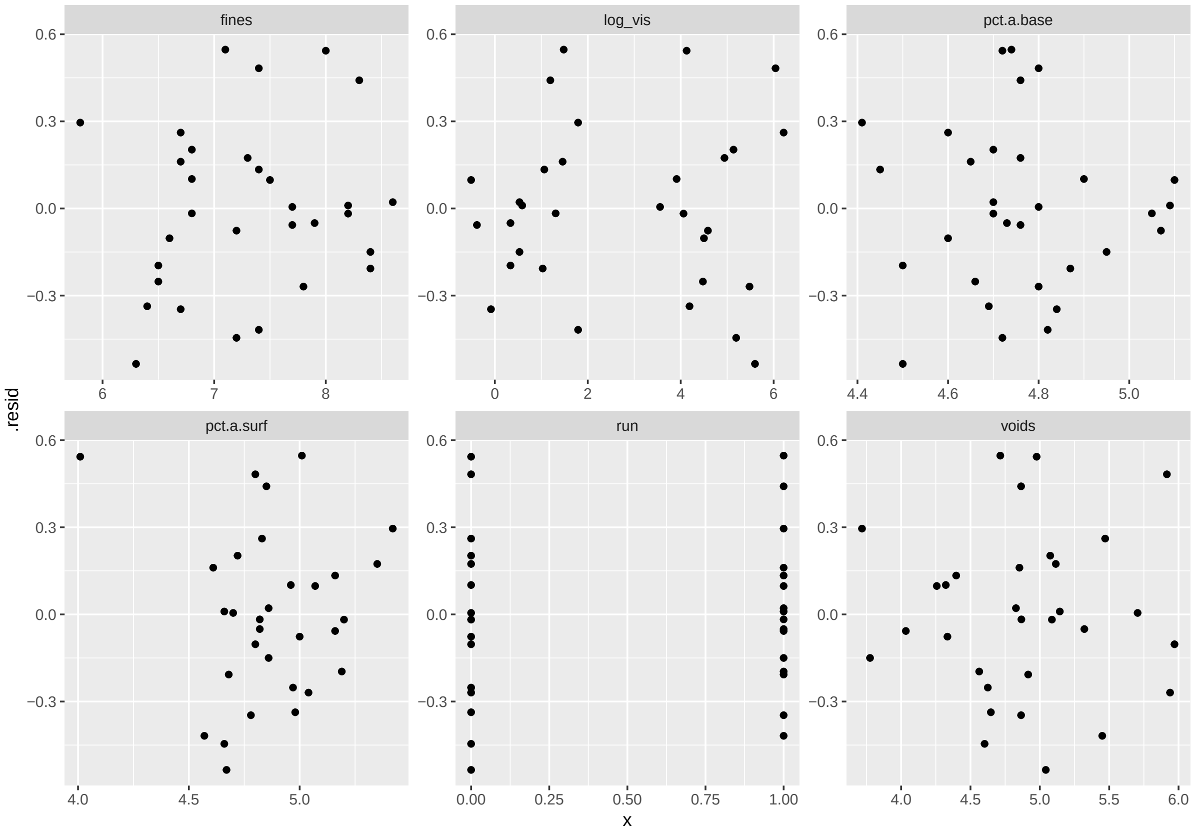
Plotting residuals against x's

- Do our trick again to put them all on one plot:

```
augment(rut.6, asphalt) %>%
  mutate(log_vis=log(viscosity)) %>%
  pivot_longer(
    c(pct.a.surf:voids, run, log_vis),
    names_to="xname", values_to="x",
  ) %>%
  ggplot(aes(y = .resid, x = x)) + geom_point() +
  facet_wrap(~xname, scales = "free") -> g2
```

Residuals against the x's

```
g2
```



Comments

- None of the plots show any sort of pattern. The points all look random on each plot.
- On the plot of fitted values (and on the one of log.viscosity), the points seem to form a "left half" and a "right half" with a gap in the middle. This is not a concern.
- One of the pct.a.surf values is low outlier (4), shows up top left of that plot.
- Only two possible values of run; the points in each group look randomly scattered around 0, with equal spreads.
- Residuals seem to go above zero further than below, suggesting a mild non-normality, but not enough to be a problem.

Variable-selection strategies

- Expert knowledge.
- Backward elimination.
- All possible regressions.
- Taking a variety of models to experts and asking their opinion.
- Use a looser cutoff to eliminate variables in backward elimination (eg. only if P-value greater than 0.20).
- If goal is prediction, eliminating worthless variables less important.
- If goal is understanding, want to eliminate worthless variables where possible.
- Results of variable selection not always reproducible, so caution advised.