Case study 3: Asphalt

The asphalt data

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

Packages for this section

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

Getting set up

```
my_url <- "http://www.utsc.utoronto.ca/~butler/c32/asphalt.txt
asphalt <- read_delim(my_url, " ")
```

```
## Parsed with column specification:
## cols(
##   pct.a.surf = col_double(),
##   pct.a.base = col_double(),
##   fines = col_double(),
##   voids = col_double(),
##   rut.depth = col_double(),
##   viscosity = col_double(),
##   run = col_double()
## )
```

- 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
```

pct.a.surf	pct.a.base	fines	voids	rut.depth	viscosity	run
4.68	4.87	8.4	4.916	6.75	2.80	1
5.19	4.50	6.5	4.563	13.00	1.40	1
4.82	4.73	7.9	5.321	14.75	1.40	1
4.85	4.76	8.3	4.865	12.60	3.30	1
4.86	4.95	8.4	3.776	8.25	1.70	1
5.16	4.45	7.4	4.397	10.67	2.90	1
4.82	5.05	6.8	4.867	7.28	3.70	1
4.86	4.70	8.6	4.828	12.67	1.70	1
4.78	4.84	6.7	4.865	12.58	0.92	1
5.16	4.76	7.7	4.034	20.60	0.68	1
4.57	4.82	7.4	5.450	3.58	6.00	1
4.61	4.65	6.7	4.853	7.00	4.30	1
5.07	5.10	7.5	4.257	26.20	0.60	1

Plotting response “rut depth” against everything else

Same idea as for plotting separate predictions on one plot:

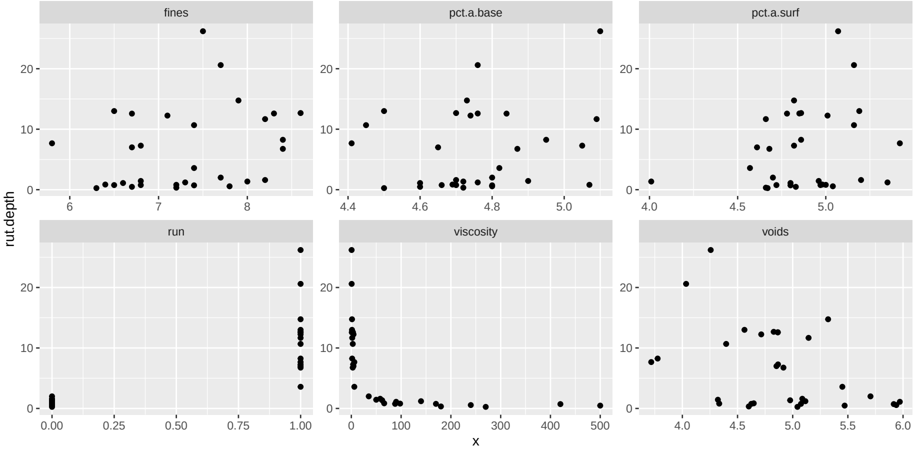
```
asphalt %>%
  pivot_longer(
    -rut.depth,
    names_to="xname", values_to="x"
  ) %>%
  ggplot(aes(x = x, y = rut.depth)) + geom_point() +
  facet_wrap(~xname, scales = "free") -> g
```

“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
```


fines
pct.a.base
pct.a.surf
run
viscosity
voids
rut.depth
x
0
10
20
6
7
8
4.4
4.6
4.8
5.0
4.0
4.5
5.0
0.00
0.25
0.50
0.75
1.00
0
100
200
300
400
500
4.0
4.5
5.0
5.5
6.0

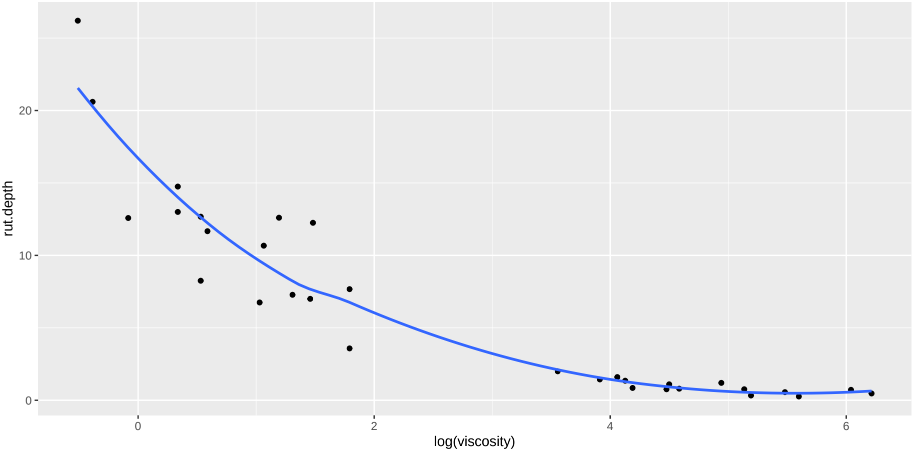
Interpreting the plots

- 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.
- 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:

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

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC
0.8060035	0.7575043	3.323526	16.61893	2e-07	7	-77.25194	170.5039

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

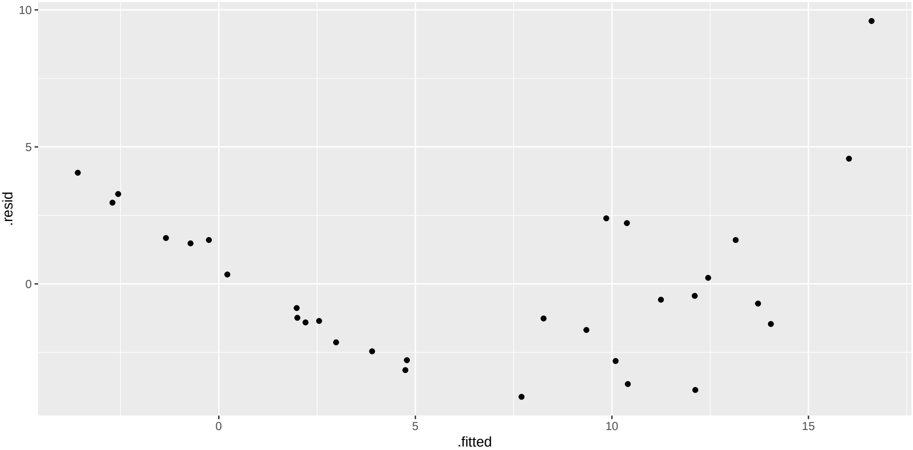
term	estimate	std.error	statistic	p.value
(Intercept)	-12.9936763	26.2187822	-0.4955866	0.6246937
pct.a.surf	3.9705671	2.4966454	1.5903608	0.1248412
pct.a.base	1.2631409	3.9702906	0.3181482	0.7531244
fines	0.1164434	1.0123885	0.1150185	0.9093873
voids	0.5892602	1.3243933	0.4449284	0.6603584
log(viscosity)	-3.1515128	0.9194468	-3.4276183	0.0022027
run	-1.9654795	3.6472048	-0.5389002	0.5949198

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?

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

```
##  [1] "pct.a.surf" "pct.a.base" "fines"
##  [4] "voids"      "rut.depth"  "viscosity"
##  [7] "run"        ".fitted"    ".se.fit"
## [10] ".resid"     ".hat"       ".sigma"
## [13] ".cooksd"    ".std.resid"
```

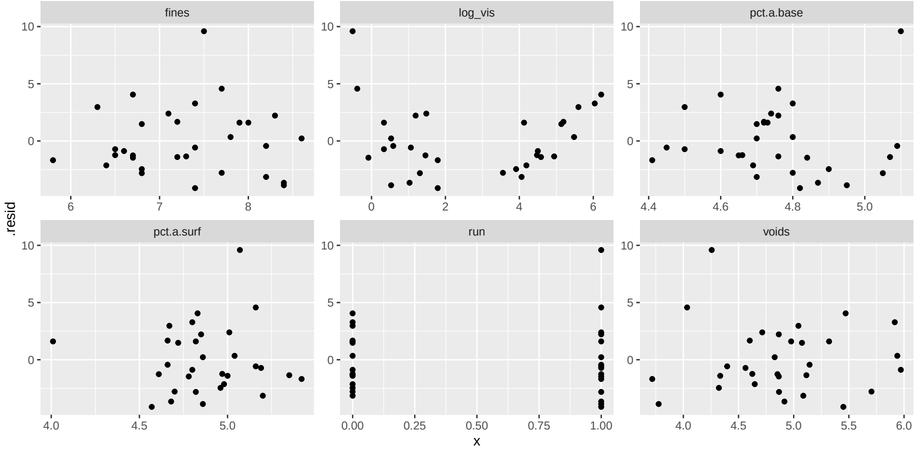
- all the stuff in original data frame, plus:
- quantities from regression (starting with a dot)

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
```


fines
log_vis
pct.a.base
pct.a.surf
run
voids
.resid
x
10
5
0
6
7
8
0
2
4
6
4.4
4.6
4.8
5.0
4.0
4.5
5.0
0.00
0.25
0.50
0.75
1.00
4.0
4.5
5.0
5.5
6.0

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.

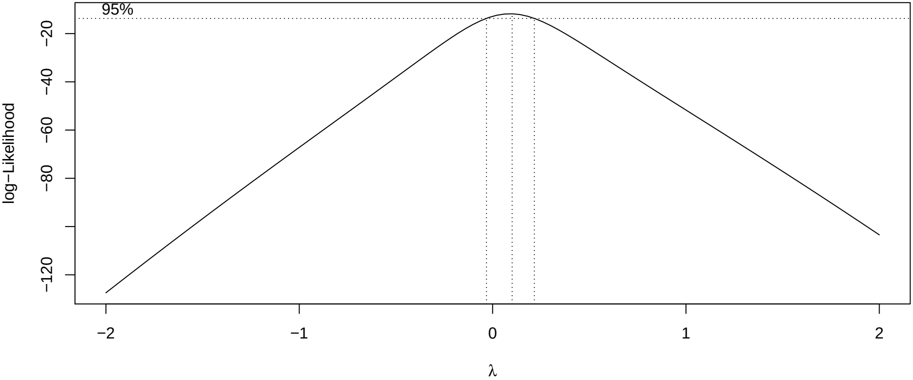
Which transformation?

- 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 ($\lambda = -1$).

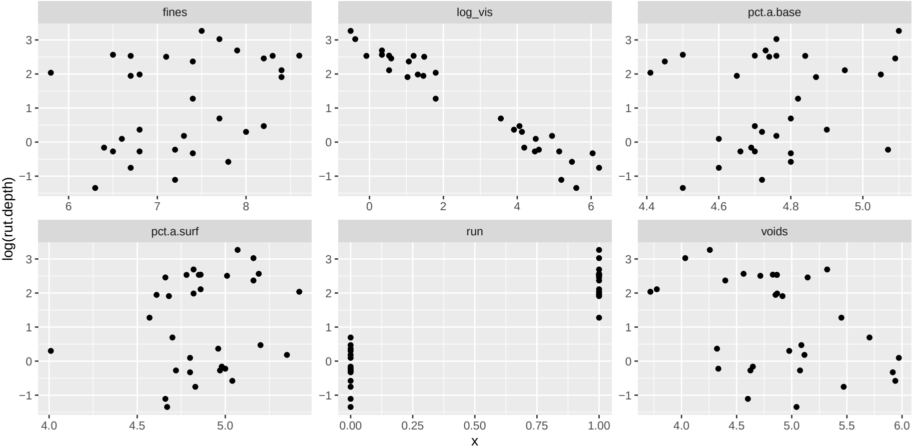
Relationships with explanatories

- 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:

```r
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

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

term	estimate	std.error	statistic	p.value
(Intercept)	-1.5729892	2.4361740	-0.6456802	0.5246124
pct.a.surf	0.5835756	0.2319811	2.5156167	0.0189818
pct.a.base	-0.1033673	0.3689080	-0.2801981	0.7817265
fines	0.0977520	0.0940682	1.0391609	0.3090864
voids	0.1988538	0.1230588	1.6159245	0.1191815
log(viscosity)	-0.5576873	0.0854324	-6.5278223	0.0000009
run	0.3400483	0.3388878	1.0034245	0.3256663

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)
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
28	2.880919	NA	NA	NA	NA
24	2.288764	4	0.5921551	1.552336	0.2191461

- 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)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-1.5729892	2.4361740	-0.6456802	0.5246124
pct.a.surf	0.5835756	0.2319811	2.5156167	0.0189818
pct.a.base	-0.1033673	0.3689080	-0.2801981	0.7817265
fines	0.0977520	0.0940682	1.0391609	0.3090864
voids	0.1988538	0.1230588	1.6159245	0.1191815
log(viscosity)	-0.5576873	0.0854324	-6.5278223	0.0000009
run	0.3400483	0.3388878	1.0034245	0.3256663

- 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:

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

term	estimate	std.error	statistic	p.value
log(viscosity)	-0.5576873	0.0854324	-6.5278223	0.0000009
pct.a.surf	0.5835756	0.2319811	2.5156167	0.0189818
voids	0.1988538	0.1230588	1.6159245	0.1191815
fines	0.0977520	0.0940682	1.0391609	0.3090864
run	0.3400483	0.3388878	1.0034245	0.3256663
(Intercept)	-1.5729892	2.4361740	-0.6456802	0.5246124
pct.a.base	-0.1033673	0.3689080	-0.2801981	0.7817265

- Largest P-value at the bottom.

Take out `pct.a.base`

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

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

term	estimate	std.error	statistic	p.value
log(viscosity)	-0.5524904	0.0818434	-6.750578	0.0000004
pct.a.surf	0.5929947	0.2252626	2.632459	0.0143218
voids	0.2004674	0.1206373	1.661737	0.1090561
(Intercept)	-2.0784987	1.6066507	-1.293684	0.2075999
run	0.3597735	0.3253286	1.105877	0.2793085
fines	0.0889458	0.0870133	1.022209	0.3164725

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

“Update”

Another way to do the same thing:

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

term	estimate	std.error	statistic	p.value
log(viscosity)	-0.5524904	0.0818434	-6.750578	0.0000004
pct.a.surf	0.5929947	0.2252626	2.632459	0.0143218
voids	0.2004674	0.1206373	1.661737	0.1090561
(Intercept)	-2.0784987	1.6066507	-1.293684	0.2075999
run	0.3597735	0.3253286	1.105877	0.2793085
fines	0.0889458	0.0870133	1.022209	0.3164725

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

Take out fines:

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

term	estimate	std.error	statistic	p.value
log(viscosity)	-0.5803887	0.0772254	-7.5155118	0.0000001
pct.a.surf	0.5483723	0.2211833	2.4792668	0.0199713
voids	0.2318797	0.1167584	1.9859780	0.0576740
run	0.2946793	0.3193108	0.9228604	0.3645654
(Intercept)	-1.2553309	1.3914683	-0.9021628	0.3752525

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)
```

term	estimate	std.error	statistic	p.value
log(viscosity)	-0.6464911	0.0287864	-22.4581853	0.0000000
pct.a.surf	0.5554686	0.2204415	2.5198000	0.0179633
voids	0.2447934	0.1155981	2.1176259	0.0435604
(Intercept)	-1.0207945	1.3643000	-0.7482185	0.4607966

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
##     -1.0207945       0.5554686       0.2447934
## log(viscosity)
##     -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:

```
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

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

model	rsq	pct.a.surf	pct.a.base	fines	voids	log.viscosity.	run
6 (1)	0.9609642	*	*	*	*	*	*
5 (1)	0.9608365	*		*	*	*	*
5 (2)	0.9593265	*	*	*	*	*	
4 (1)	0.9591996	*			*	*	*
4 (2)	0.9589206	*		*	*	*	
3 (1)	0.9578631	*			*	*	
3 (2)	0.9534561	*		*		*	
2 (1)	0.9508647	*				*	
2 (2)	0.9479541				*	*	
1 (1)	0.9452562					*	
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
3 (1)	0.9531812	*			*	*	
5 (1)	0.9530038	*		*	*	*	*
4 (1)	0.9529226	*			*	*	*
4 (2)	0.9526007	*		*	*	*	
6 (1)	0.9512052	*	*	*	*	*	*
5 (2)	0.9511918	*	*	*	*	*	
3 (2)	0.9482845	*		*		*	
2 (1)	0.9473550	*				*	
2 (2)	0.9442365				*	*	
1 (1)	0.9433685					*	
1 (2)	0.8576662						*

Revisiting the best model

- Best model was our rut.6:

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

term	estimate	std.error	statistic	p.value
(Intercept)	-1.0207945	1.3643000	-0.7482185	0.4607966
pct.a.surf	0.5554686	0.2204415	2.5198000	0.0179633
voids	0.2447934	0.1155981	2.1176259	0.0435604
log(viscosity)	-0.6464911	0.0287864	-22.4581853	0.0000000

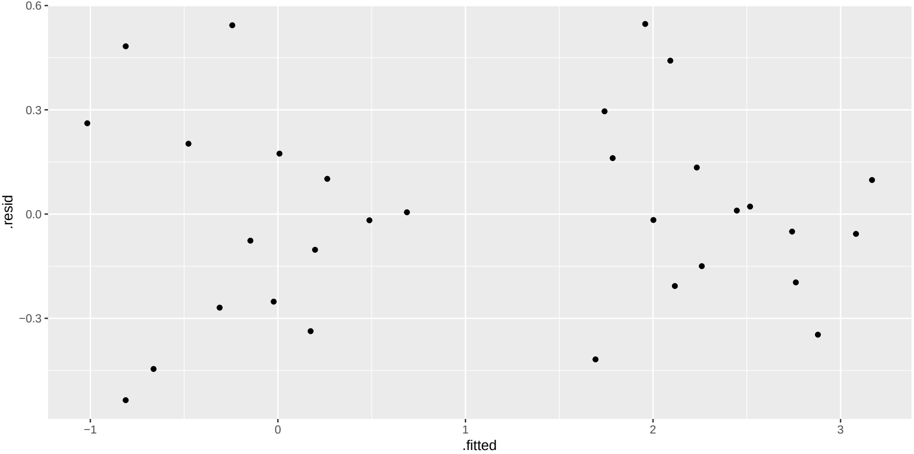
Revisiting (2)

- 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
```



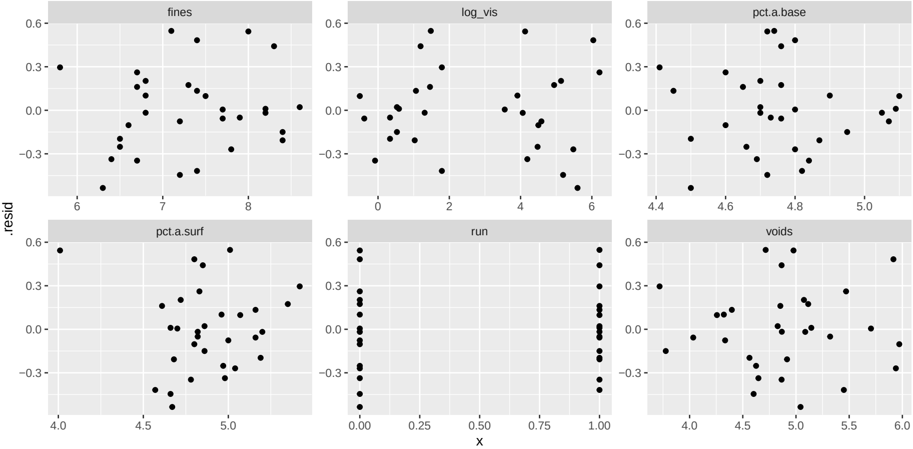
Plotting residuals against x's

- 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.