432 Class 05

https://thomaselove.github.io/432-2024/

2024-01-30

## Today’s Agenda

* The HELP study (today’s main data) and preliminaries
* Using ols to fit a linear model
  + Obtaining coefficients and basic summaries, ANOVA in ols
  + Validating summary statistics like
  + Plot Effects with summary and Predict
  + Building and using a nomogram
  + Evaluating Calibration
  + Influential points and dfbeta
* Building Non-Linear Predictors in ols
  + Polynomial Functions
  + Restricted Cubic Splines
* Appendix: Additional Spline Examples

## Today’s R Setup

knitr::opts\_chunk$set(comment = NA)  
  
library(mosaic) ## auto-loads mosaicData  
library(janitor)  
library(gt)  
library(broom)  
library(rsample)  
library(yardstick)  
library(patchwork)  
library(GGally) ## for scatterplot matrix  
library(rms) ## auto-loads Hmisc  
library(tidyverse)  
  
theme\_set(theme\_bw())

# Data from the HELP study

## New Data (helpdat: HELP study)

Today’s main data set comes from the Health Evaluation and Linkage to Primary Care trial, and is stored as HELPrct in the mosaicData package.

HELP was a clinical trial of adult inpatients recruited from a detoxification unit. Patients with no primary care physician were randomized to receive a multidisciplinary assessment and a brief motivational intervention or usual care, with the goal of linking them to primary medical care.

## Key Variables for Today

| Variable | Description |
| --- | --- |
| id | subject identifier |
| cesd | Center for Epidemiologic Studies Depression measure (higher scores indicate more depressive symptoms) |
| age | subject age (in years) |
| sex | female (n = 107) or male (n = 346) |
| subst | primary substance of abuse (alcohol, cocaine or heroin) |
| mcs | SF-36 Mental Component Score (lower = worse status) |
| pcs | SF-36 Physical Component Score (lower = worse status) |
| pss\_fr | perceived social support by friends (higher = more support) |

* All measures from baseline during the subjects’ detoxification stay.
* More data and details at https://nhorton.people.amherst.edu/help/.

## The helpdat data

We will look at 453 subjects with complete data today.

helpdat <- tibble(mosaicData::HELPrct) |>  
 select(id, cesd, age, sex, subst = substance,  
 mcs, pcs, pss\_fr)  
  
df\_stats(~ cesd + age + mcs + pcs + pss\_fr, data = helpdat) |>  
 gt() |> fmt\_number(min:sd, decimals = 2) |>  
 tab\_options(table.font.size = 20)

| response | min | Q1 | median | Q3 | max | mean | sd | n | missing |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| cesd | 1.00 | 25.00 | 34.00 | 41.00 | 60.00 | 32.85 | 12.51 | 453 | 0 |
| age | 19.00 | 30.00 | 35.00 | 40.00 | 60.00 | 35.65 | 7.71 | 453 | 0 |
| mcs | 6.76 | 21.68 | 28.60 | 40.94 | 62.18 | 31.68 | 12.84 | 453 | 0 |
| pcs | 14.07 | 40.38 | 48.88 | 56.95 | 74.81 | 48.05 | 10.78 | 453 | 0 |
| pss\_fr | 0.00 | 3.00 | 7.00 | 10.00 | 14.00 | 6.71 | 4.00 | 453 | 0 |

## helpdat categorical variables

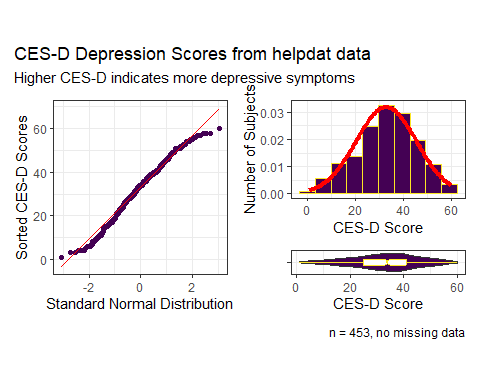
helpdat |> tabyl(sex, subst) |>   
 adorn\_totals(where = c("row", "col")) |>  
 adorn\_percentages(denominator = "row") |>  
 adorn\_pct\_formatting() |>  
 adorn\_ns(position = "front") |>  
 adorn\_title(placement = "combined") |>  
 gt() |> tab\_options(table.font.size = 20)

| sex/subst | alcohol | cocaine | heroin | Total |
| --- | --- | --- | --- | --- |
| female | 36 (33.6%) | 41 (38.3%) | 30 (28.0%) | 107 (100.0%) |
| male | 141 (40.8%) | 111 (32.1%) | 94 (27.2%) | 346 (100.0%) |
| Total | 177 (39.1%) | 152 (33.6%) | 124 (27.4%) | 453 (100.0%) |

## Our Outcome (CES-Depression score)

p1 <- ggplot(helpdat, aes(sample = cesd)) +  
 geom\_qq(col = '#440154') + geom\_qq\_line(col = "red") +   
 theme(aspect.ratio = 1) +  
 labs(y = "Sorted CES-D Scores",   
 x = "Standard Normal Distribution")  
  
p2 <- ggplot(helpdat, aes(x = cesd)) +  
 geom\_histogram(aes(y = stat(density)),   
 bins = 10, fill = '#440154', col = '#FDE725') +  
 stat\_function(fun = dnorm,   
 args = list(mean = mean(helpdat$cesd),   
 sd = sd(helpdat$cesd)),  
 col = "red", lwd = 1.5) +  
 labs(y = "Number of Subjects", x = "CES-D Score")  
  
p3 <- ggplot(helpdat, aes(x = cesd, y = "")) +  
 geom\_violin(fill = '#440154') +  
 geom\_boxplot(width = 0.3, col = '#FDE725', notch = TRUE,   
 outlier.color = '#440154') +  
 labs(x = "CES-D Score", y = "")  
  
p1 + (p2 / p3 + plot\_layout(heights = c(4,1))) +  
 plot\_annotation(title = "CES-D Depression Scores from helpdat data",  
 subtitle = "Higher CES-D indicates more depressive symptoms",  
 caption = "n = 453, no missing data")

Warning: `stat(density)` was deprecated in ggplot2 3.4.0.  
ℹ Please use `after\_stat(density)` instead.



## Describing our outcome CES-D

describe(helpdat$cesd)

helpdat$cesd : CESD at baseline   
 n missing distinct Info Mean Gmd .05 .10   
 453 0 58 0.999 32.85 14.23 10.0 15.2   
 .25 .50 .75 .90 .95   
 25.0 34.0 41.0 49.0 52.4   
  
lowest : 1 3 4 5 6, highest: 55 56 57 58 60

* Info measures the variable’s information between 0 and 1: the higher the Info, the more continuous the variable is (the fewer ties there are.)
* Gmd = Gini’s mean difference, a robust measure of variation. If you randomly selected two of the 453 subjects many times, the mean difference in cesd would be 14.23 points.

## We have some labels in our data

str(helpdat)

tibble [453 × 8] (S3: tbl\_df/tbl/data.frame)  
 $ id : int [1:453] 1 2 3 4 5 6 7 8 9 10 ...  
 ..- attr(\*, "label")= chr "subject ID"  
 $ cesd : int [1:453] 49 30 39 15 39 6 52 32 50 46 ...  
 ..- attr(\*, "label")= chr "CESD at baseline"  
 $ age : int [1:453] 37 37 26 39 32 47 49 28 50 39 ...  
 ..- attr(\*, "label")= chr "age (years)"  
 $ sex : Factor w/ 2 levels "female","male": 2 2 2 1 2 1 1 2 1 2 ...  
 ..- attr(\*, "label")= chr "sex"  
 $ subst : Factor w/ 3 levels "alcohol","cocaine",..: 2 1 3 3 2 2 2 1 1 3 ...  
 ..- attr(\*, "label")= chr "primary substance of abuse"  
 $ mcs : num [1:453] 25.11 26.67 6.76 43.97 21.68 ...  
 ..- attr(\*, "label")= chr "SF-36 Mental Component Score"  
 $ pcs : num [1:453] 58.4 36 74.8 61.9 37.3 ...  
 ..- attr(\*, "label")= chr "SF-36 Physical Component Score"  
 $ pss\_fr: int [1:453] 0 1 13 11 10 5 1 4 5 0 ...  
 ..- attr(\*, "label")= chr "perceived social support by friends"

## Scatterplot Matrix (code)

temp <- helpdat |>  
 select(age, mcs, pcs, pss\_fr, sex, subst, cesd)  
  
ggpairs(temp) ## ggpairs from the GGally package

We place the outcome (cesd) last (result on next slide.)

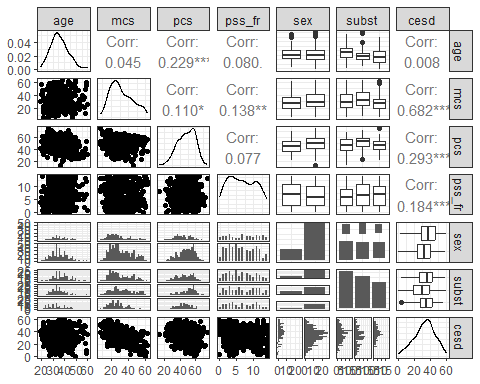
### Saving the Data Set

write\_rds(helpdat, "c05/data/helpdat.Rds")

## Scatterplot Matrix (result)

temp <- helpdat |>  
 select(age, mcs, pcs, pss\_fr, sex, subst, cesd)  
  
ggpairs(temp) ## ggpairs from the GGally package

`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# Using ols to fit a linear regression model

## Fitting using ols

The ols function stands for ordinary least squares and comes from the rms package, by Frank Harrell and colleagues. Any model fit with lm can also be fit with ols.

* To predict var\_y using var\_x from the my\_tibble data, we would use the following syntax:

dd <- datadist(my\_tibble)  
options(datadist = "dd")  
  
model\_name <- ols(var\_y ~ var\_x, data = my\_tibble,  
 x = TRUE, y = TRUE)

This leaves a few questions…

## What’s the datadist stuff doing?

Before fitting an ols model to data from my\_tibble, use:

dd <- datadist(my\_tibble)  
options(datadist = "dd")

Run (the datadist code above) once before any models are fitted, storing the distribution summaries for all potential variables. Adjustment values are 0 for binary variables, the most frequent category (or optionally the first category level) for categorical (factor) variables, the middle level for ordered factor variables, and medians for continuous variables. (excerpt from datadist documentation)

## Why use x = TRUE, y = TRUE?

Once we’ve set up the summaries with datadist, we fit a model:

model\_name <- ols(var\_y ~ var\_x, data = my\_tibble,  
 x = TRUE, y = TRUE)

* ols stores additional information beyond what lm does
* x = TRUE and y = TRUE save even more expanded information for building plots and summarizing fit.
* The defaults are x = FALSE, y = FALSE, but in 432, we’ll want them saved.

## Using ols to fit a model

Let’s try to predict our outcome (cesd) using mcs and subst

* Start with setting up the datadist
* Then fit the model, including x = TRUE, y = TRUE

dd <- datadist(helpdat)  
options(datadist = "dd")  
  
mod1 <- ols(cesd ~ mcs + subst, data = helpdat,  
 x = TRUE, y = TRUE)

## Contents of mod1?

mod1

Linear Regression Model  
  
ols(formula = cesd ~ mcs + subst, data = helpdat, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 295.10 R2 0.479   
sigma9.0657 d.f. 3 R2 adj 0.475   
d.f. 449 Pr(> chi2) 0.0000 g 9.827   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-25.43696 -6.74592 0.09334 6.16212 24.24842   
  
 Coef S.E. t Pr(>|t|)  
Intercept 55.3026 1.2724 43.46 <0.0001   
mcs -0.6570 0.0337 -19.48 <0.0001   
subst=cocaine -3.4440 1.0055 -3.43 0.0007   
subst=heroin -1.7791 1.0681 -1.67 0.0965

## New elements in ols

For our mod1,

* Model Likelihood Ratio test output includes LR chi2 = 295.10, d.f. = 3, Pr(> chi2) = 0.0000

The log of the likelihood ratio, multiplied by -2, yields a test against a distribution. Interpret this as a goodness-of-fit test that compares mod1 to a null model with only an intercept term. In ols this is similar to a global (ANOVA) F test.

## New elements in ols

Under the values, we have g = 9.827.

* This is the -index, based on Gini’s mean difference. If you randomly selected two of the subjects in the model, the average difference in predicted cesd will be 9.827.
* This can be compared to the Gini’s mean difference for the original cesd values, from describe, which was Gmd = 14.23.

## Validate summaries from an ols fit

* Can we validate summary statistics by resampling?

set.seed(432)  
validate(mod1)

index.orig training test optimism index.corrected n  
R-square 0.4787 0.4874 0.4737 0.0137 0.4650 40  
MSE 81.4606 79.7851 82.2361 -2.4510 83.9116 40  
g 9.8272 9.9133 9.8038 0.1095 9.7177 40  
Intercept 0.0000 0.0000 0.2793 -0.2793 0.2793 40  
Slope 1.0000 1.0000 0.9894 0.0106 0.9894 40

* The data used to fit the model provide an over-optimistic view of the quality of fit.
* We’re interested here in assessing how well the model might work in new data, using a resampling approach.

## Interpreting Resampling Validation

| – | index.orig | training | test | optimism | index.corrected | n |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0.4787 | 0.4874 | 0.4737 | 0.0137 | 0.4650 | 40 |

* index.orig for is 0.4787. That’s what we get from the data used to fit mod1.
* With validate we create 40 (by default) bootstrapped resamples of the data and then split each of those into training and test samples.
  + For each of the 40 splits, R refits the model (same predictors) in the training sample to obtain : mean across 40 splits is 0.4874
  + Check each model in its test sample: average was 0.4737
* optimism = training result - test result = 0.0137
* index.corrected = index.orig - optimism = 0.4650

While our *nominal* is 0.4787; correcting for optimism yields *validated* of 0.4650, so we conclude that = 0.4650 better estimates how mod1 will perform in new data.

## ANOVA for mod1 fit by ols

anova(mod1)

Analysis of Variance Response: cesd   
  
 Factor d.f. Partial SS MS F P   
 mcs 1 31182.7237 31182.72373 379.42 <.0001  
 subst 2 968.7563 484.37816 5.89 0.003   
 REGRESSION 3 33886.8359 11295.61195 137.44 <.0001  
 ERROR 449 36901.6542 82.18631

* This adds a line for the complete regression model (both terms) which can be helpful, but is otherwise the same as anova() after a fit using lm().
* As with lm, this is a sequential ANOVA table, so if we had included subst in the model first, we’d get a different SS, MS, F and p for mcs and subst, but the same REGRESSION and ERROR results.

## summary for mod1 fit by ols

summary(mod1, conf.int = 0.90)

Effects Response : cesd   
  
 Factor Low High Diff. Effect S.E. Lower 0.9  
 mcs 21.676 40.941 19.266 -12.6580 0.64984 -13.7290   
 subst - cocaine:alcohol 1.000 2.000 NA -3.4440 1.00550 -5.1013   
 subst - heroin:alcohol 1.000 3.000 NA -1.7791 1.06810 -3.5396   
 Upper 0.9   
 -11.587000  
 -1.786700  
 -0.018654

* How do we interpret the subst effects estimated by this model?
  + Effect of subst being cocaine instead of alcohol on ces\_d is -3.44 assuming no change in mcs, with 90% CI (-5.10, -1.79).
  + Effect of subst being heroin instead of alcohol on ces\_d is -1.78 assuming no change in mcs, with 90% CI (-3.54, -0.02).

But what about the mcs effect?

## summary for mod1 fit by ols

summary(mod1, conf.int = 0.90)

Effects Response : cesd   
  
 Factor Low High Diff. Effect S.E. Lower 0.9  
 mcs 21.676 40.941 19.266 -12.6580 0.64984 -13.7290   
 subst - cocaine:alcohol 1.000 2.000 NA -3.4440 1.00550 -5.1013   
 subst - heroin:alcohol 1.000 3.000 NA -1.7791 1.06810 -3.5396   
 Upper 0.9   
 -11.587000  
 -1.786700  
 -0.018654

* Effect of mcs: -12.66 is the estimated change in cesd associated with a move from mcs = 21.68 (see Low value) to mcs = 40.94 (the High value) assuming no change in subst.
* ols chooses the Low and High values from the interquartile range.

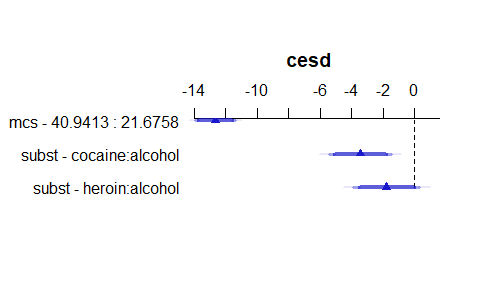
quantile(helpdat$mcs, c(0.25, 0.75))

25% 75%   
21.67575 40.94134

## Plot the summary to see effect sizes

* Goal: plot effect sizes for similar moves within predictor distributions.

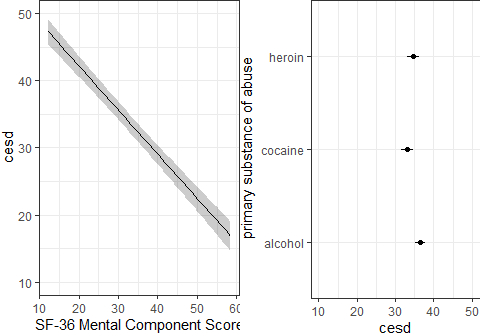
plot(summary(mod1))



* The triangles indicate the point estimate, augmented with confidence interval bars.
  + The 90% confidence intervals are plotted with the thickest bars.
  + The 95% CIs are then shown with thinner, more transparent bars.
  + Finally, the 99% CIs are shown as the longest, thinnest bars.

## Plot the individual effects?

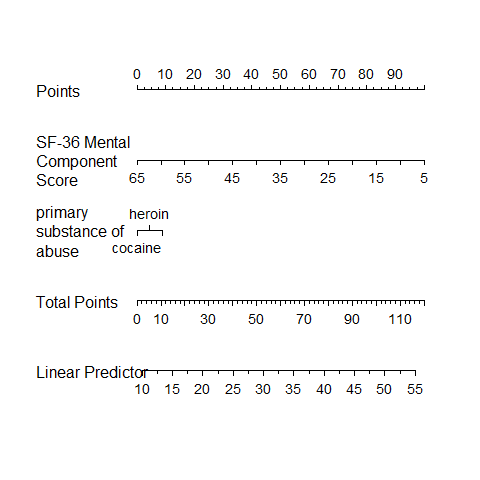
ggplot(Predict(mod1, conf.int = 0.95), layout = c(1,2))



* At left, impact of changing mcs on cesd holding subst at its baseline (alcohol).
* At right, impact of changing subst on cesd holding mcs at its median (28.602417).
* Defaults: add 95% CI bands and layout tries for a square.

## Build a nomogram for the ols fit

plot(nomogram(mod1))



## Nomograms

For complex models (this model isn’t actually very complex) it can be helpful to have a tool that will help you see the modeled effects in terms of their impact on the predicted outcome.

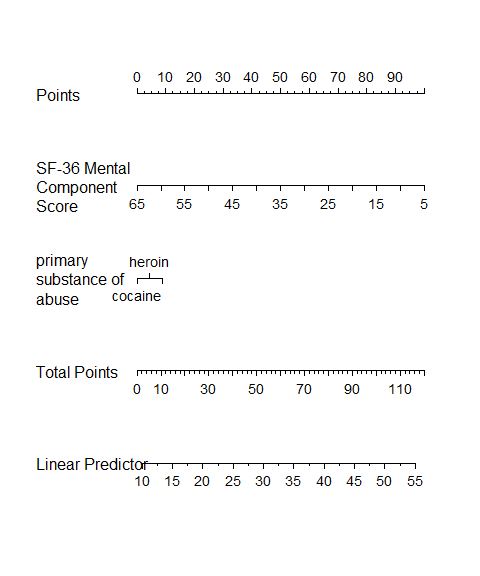
A *nomogram* is an established graphical tool for doing this.

* Find the value of each predictor on its provided line, and identify the “points” for that predictor by drawing a vertical line up to the “Points”.
* Then sum up the points over all predictors to obtain “Total Points”.
* Draw a vertical line down from the “Total Points” to the “Linear Predictor” to get the predicted cesd for this subject.

## Using the nomogram for the mod1 fit

Predicted cesd if mcs = 35 and subst = heroin?

plot(nomogram(mod1))



## Actual Prediction for this subject…

* The predict function for our ols fit provides fitted values.

predict(mod1, newdata = tibble(mcs = 35, subst = "heroin"))

1   
30.52766

* The broom package can also support rms fits

augment(mod1, newdata = tibble(mcs = 35, subst = "heroin"))

Warning: The `augment()` method for objects of class `ols` is not maintained by the broom team, and is only supported through the `lm` tidier method. Please be cautious in interpreting and reporting broom output.  
  
This warning is displayed once per session.

# A tibble: 1 × 3  
 mcs subst .fitted  
 <dbl> <chr> <dbl>  
1 35 heroin 30.5

## Assessing the Calibration of mod1

We would like our model to be well-calibrated, in the following sense…

* Suppose our model assigns a predicted outcome of 6 to several subjects. If the model is well-calibrated, this means we expect the mean of those subjects’ actual outcomes to be very close to 6.

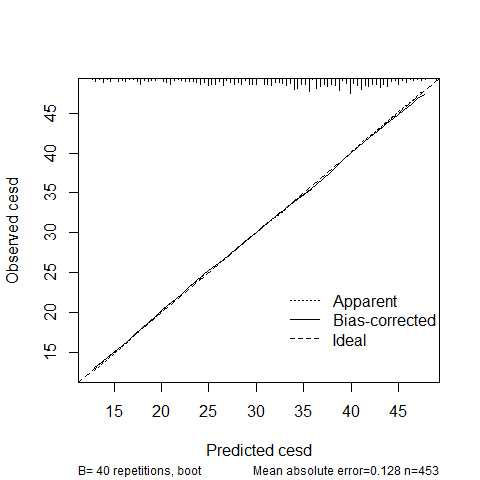
## Assessing the Calibration of mod1

We’d like to look at the relationship between the observed cesd outcome and our predicted cesd from the model.

* The calibration plot we’ll create provides two estimates (with and without bias-correction) of the predicted vs. observed values of our outcome, and compares these to the ideal scenario (predicted = observed).
* The plot uses resampling validation to produce bias-corrected estimates and uses lowess smooths to connect across predicted values.
* Calibration plots require x = TRUE, y = TRUE in ols.

## Calibration Plot for mod1

set.seed(43299); plot(calibrate(mod1))



n=453 Mean absolute error=0.128 Mean squared error=0.02428  
0.9 Quantile of absolute error=0.267

## Influential Points for mod1?

The dfbeta value for a particular subject and coefficient is the change in the coefficient that happens when the subject is excluded from the model.

which.influence(mod1, cutoff = 0.2)

$Intercept  
[1] "8" "351" "405" "433"  
  
$mcs  
[1] "351" "402" "450"  
  
$subst  
[1] "351"

* These are the subjects that have absolute values of dfbetas that exceed the specified cutoff (default is 0.2 but it’s an arbitrary choice.)

## Show influential points directly?

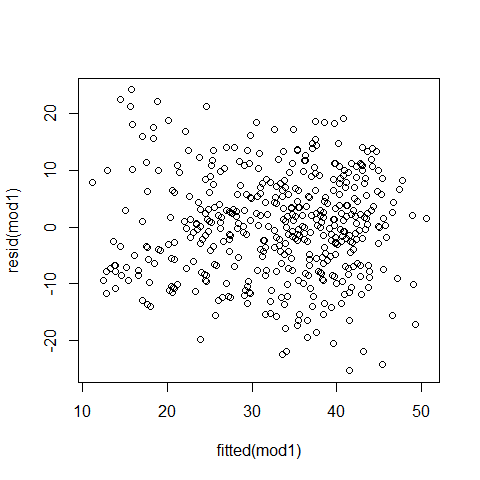
w <- which.influence(mod1, cutoff = 0.2)  
d <- helpdat |> select(mcs, subst, cesd) |> data.frame()  
show.influence(w, d)

Count mcs subst  
8 1 9.16053 alcohol  
351 3 \*57.48944 \*heroin   
402 1 \*55.47938 alcohol  
405 1 15.07887 alcohol  
433 1 18.59431 alcohol  
450 1 \*62.17550 alcohol

* Count = number of coefficients where this row appears influential.
* Use helpdat |> slice(351) to see row 351 in its entirety.
* Use residual plots (with an lm fit) to check Cook’s distances.

## Residuals vs. Fitted plot from ols

plot(resid(mod1) ~ fitted(mod1))



## Fitting all Residual Plots for mod1

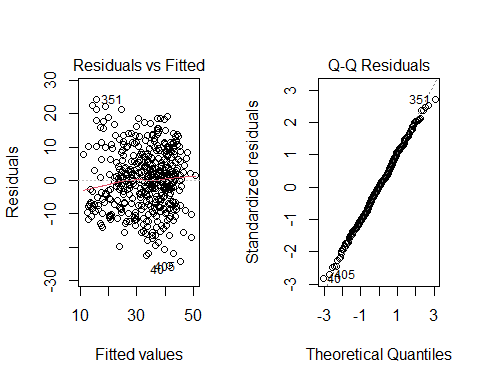
To fit more complete residual plots (and other things) we can fit the lm version of this same model…

mod1\_lm <- lm(cesd ~ mcs + subst, data = helpdat)  
  
par(mfrow = c(2,2)); plot(mod1\_lm); par(mfrow = c(1,1))

* Plots are shown on the next two slides. While the subject in row 351 is more influential than most other points, it doesn’t reach the standard of a problematic Cook’s distance.

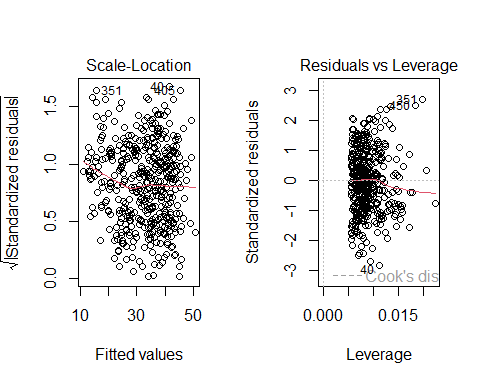
## First Two mod1\_lm Residual Plots

mod1\_lm <- lm(cesd ~ mcs + subst, data = helpdat)  
  
par(mfrow = c(1,2)); plot(mod1\_lm, which = c(1,2))



## Second Two mod1\_lm Residual Plots

par(mfrow = c(1,2)); plot(mod1\_lm, which = c(3,5))



# Non-Linear Terms: Polynomials

## Non-Linear Terms

In building a linear regression model, we’re most often going to be thinking about:

* for quantitative predictors, some curvature…
  + perhaps polynomial terms
  + but more often restricted cubic splines
* for any predictors, possible interactions
  + between categorical predictors
  + between categorical and quantitative predictors
  + between quantitative predictors

## Polynomial Regression

A polynomial regression involves a polynomial in the variable x of degree D (linear combination of powers of x up to D.)

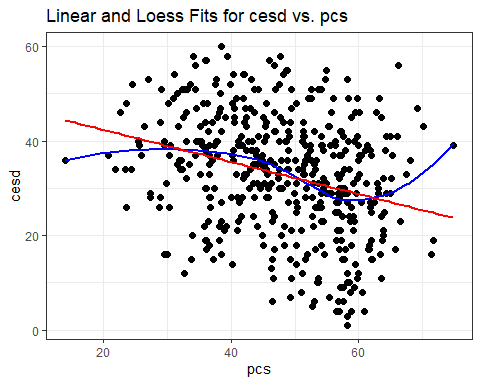
* Linear:
* Quadratic:
* Cubic:
* Quartic:

An **orthogonal polynomial** sets up a model design matrix and then scales those columns so that each column is uncorrelated with the others.

## Use pcs to predict cesd?

* Let’s look at both a linear fit and a loess smooth to see if they indicate meaningfully different things about the association between pcs and cesd

ggplot(helpdat, aes(x = pcs, y = cesd)) +   
 geom\_point(size = 2) +  
 geom\_smooth(method = "loess", formula = y ~ x,   
 se = FALSE, col = "blue") +  
 geom\_smooth(method = "lm", formula = y ~ x,  
 se = FALSE, col = "red") +   
 labs(title = "Linear and Loess Fits for cesd vs. pcs")



## Polynomial regression with ols

dd <- datadist(helpdat)  
options(datadist = "dd")  
  
mod\_B1 <- ols(cesd ~ pcs,   
 data = helpdat, x = TRUE, y = TRUE)  
mod\_B2 <- ols(cesd ~ pol(pcs, 2),   
 data = helpdat, x = TRUE, y = TRUE)  
mod\_B3 <- ols(cesd ~ pol(pcs, 3),  
 data = helpdat, x = TRUE, y = TRUE)

* Note the use of pol() from the rms package here to fit orthogonal polynomials, rather than poly() which we used in an lm fit.

## Model B1 (linear in pcs)

mod\_B1

Linear Regression Model  
  
ols(formula = cesd ~ pcs, data = helpdat, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 40.57 R2 0.086   
sigma11.9796 d.f. 1 R2 adj 0.084   
d.f. 451 Pr(> chi2) 0.0000 g 4.177   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-28.4116 -7.8036 0.6846 8.7917 29.3281   
  
 Coef S.E. t Pr(>|t|)  
Intercept 49.1673 2.5728 19.11 <0.0001   
pcs -0.3396 0.0522 -6.50 <0.0001

## Model B2 (quadratic poly. in pcs)

mod\_B2

Linear Regression Model  
  
ols(formula = cesd ~ pol(pcs, 2), data = helpdat, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 40.68 R2 0.086   
sigma11.9915 d.f. 2 R2 adj 0.082   
d.f. 450 Pr(> chi2) 0.0000 g 4.199   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-28.387 -7.750 0.591 8.634 29.697   
  
 Coef S.E. t Pr(>|t|)  
Intercept 46.4007 8.7967 5.27 <0.0001   
pcs -0.2136 0.3867 -0.55 0.5809   
pcs^2 -0.0014 0.0041 -0.33 0.7424

## Model B3 (cubic polynomial in pcs)

mod\_B3

Linear Regression Model  
  
ols(formula = cesd ~ pol(pcs, 3), data = helpdat, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 48.70 R2 0.102   
sigma11.8991 d.f. 3 R2 adj 0.096   
d.f. 449 Pr(> chi2) 0.0000 g 4.556   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-27.5245 -8.2651 0.7988 8.9004 27.4480   
  
 Coef S.E. t Pr(>|t|)  
Intercept -13.4076 22.8605 -0.59 0.5578   
pcs 4.1323 1.5825 2.61 0.0093   
pcs^2 -0.1010 0.0354 -2.85 0.0046   
pcs^3 0.0007 0.0003 2.83 0.0049

## Store the polynomial fits

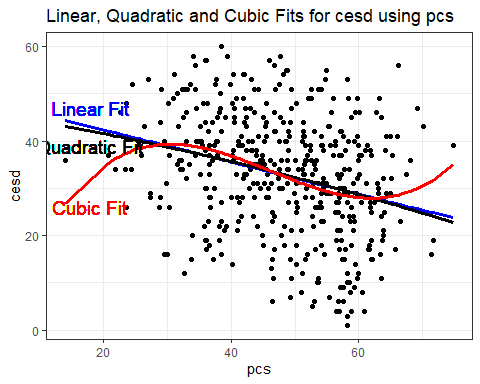
First, we need to store the values. Since broom doesn’t play well with ols fits, so I’ll just add the predictions as columns

cesd\_fits <- helpdat |>  
 mutate(fitB1 = predict(mod\_B1),  
 fitB2 = predict(mod\_B2),  
 fitB3 = predict(mod\_B3))

## Plot the polynomial fits

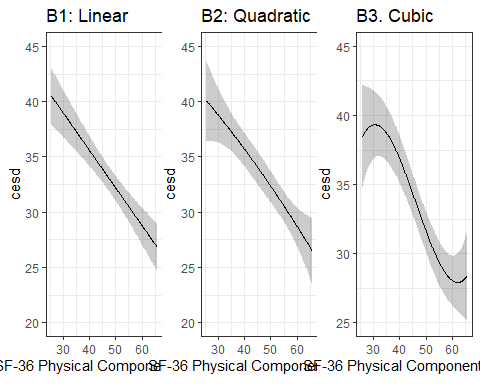
ggplot(cesd\_fits, aes(x = pcs, y = cesd)) +  
 geom\_point() +  
 geom\_line(aes(x = pcs, y = fitB1),  
 col = "blue", size = 1.25) +  
 geom\_line(aes(x = pcs, y = fitB2),  
 col = "black", size = 1.25) +  
 geom\_line(aes(x = pcs, y = fitB3),  
 col = "red", size = 1.25) +  
 geom\_text(x = 18, y = 47, label = "Linear Fit",   
 size = 5, col = "blue") +  
 geom\_text(x = 18, y = 39, label = "Quadratic Fit",   
 size = 5, col = "black") +  
 geom\_text(x = 18, y = 26, label = "Cubic Fit",   
 size = 5, col = "red") +  
 labs(title = "Linear, Quadratic and Cubic Fits for cesd using pcs")

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
ℹ Please use `linewidth` instead.



## Plot polynomial fits with Predict()

p1 <- ggplot(Predict(mod\_B1)) + ggtitle("B1: Linear")  
p2 <- ggplot(Predict(mod\_B2)) + ggtitle("B2: Quadratic")  
p3 <- ggplot(Predict(mod\_B3)) + ggtitle("B3. Cubic")  
  
p1 + p2 + p3



# Non-Linear Terms: Splines

## Types of Splines

* A **linear spline** is a continuous function formed by connecting points (called **knots** of the spline) by line segments.
* A **restricted cubic spline** is a way to build highly complicated curves into a regression equation in a fairly easily structured way.
* A restricted cubic spline is a series of polynomial functions joined together at the knots.
  + Such a spline gives us a way to flexibly account for non-linearity without over-fitting the model.

## How complex should our spline be?

Restricted cubic splines can fit many different types of non-linearities. Specifying the number of knots is all you need to do in R to get a reasonable result from a restricted cubic spline.

The most common choices are 3, 4, or 5 knots.

* 3 Knots, 2 degrees of freedom, allows the curve to “bend” once.
* 4 Knots, 3 degrees of freedom, lets the curve “bend” twice.
* 5 Knots, 4 degrees of freedom, lets the curve “bend” three times.

## Restricted Cubic Splines with ols

Let’s consider a restricted cubic spline model for cesd based on pcs with:

* 3 knots in modC3, 4 knots in modC4, and 5 knots in modC5

dd <- datadist(helpdat)  
options(datadist = "dd")  
  
mod\_C3 <- ols(cesd ~ rcs(pcs, 3),   
 data = helpdat, x = TRUE, y = TRUE)  
mod\_C4 <- ols(cesd ~ rcs(pcs, 4),   
 data = helpdat, x = TRUE, y = TRUE)  
mod\_C5 <- ols(cesd ~ rcs(pcs, 5),  
 data = helpdat, x = TRUE, y = TRUE)

## Model C3 (3-knot spline in pcs)

mod\_C3

Linear Regression Model  
  
ols(formula = cesd ~ rcs(pcs, 3), data = helpdat, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 40.79 R2 0.086   
sigma11.9901 d.f. 2 R2 adj 0.082   
d.f. 450 Pr(> chi2) 0.0000 g 4.206   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-28.3462 -7.7005 0.5098 8.6376 29.8454   
  
 Coef S.E. t Pr(>|t|)  
Intercept 47.3631 4.7053 10.07 <0.0001   
pcs -0.2908 0.1187 -2.45 0.0146   
pcs' -0.0624 0.1363 -0.46 0.6471

## Model C4 (4-knot spline in pcs)

mod\_C4

Linear Regression Model  
  
ols(formula = cesd ~ rcs(pcs, 4), data = helpdat, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 51.31 R2 0.107   
sigma11.8648 d.f. 3 R2 adj 0.101   
d.f. 449 Pr(> chi2) 0.0000 g 4.590   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-28.3147 -8.2830 0.8559 8.8866 26.5458   
  
 Coef S.E. t Pr(>|t|)  
Intercept 33.3298 6.5742 5.07 <0.0001   
pcs 0.1464 0.1856 0.79 0.4308   
pcs' -1.4383 0.4497 -3.20 0.0015   
pcs'' 6.2561 1.9076 3.28 0.0011

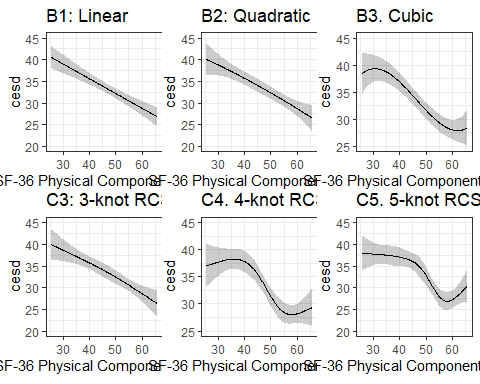
## Model C5 (5-knot spline in pcs)

mod\_C5

Linear Regression Model  
  
ols(formula = cesd ~ rcs(pcs, 5), data = helpdat, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 54.64 R2 0.114   
sigma11.8345 d.f. 4 R2 adj 0.106   
d.f. 448 Pr(> chi2) 0.0000 g 4.744   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-29.396 -7.928 1.016 8.762 26.974   
  
 Coef S.E. t Pr(>|t|)  
Intercept 39.0631 7.8282 4.99 <0.0001   
pcs -0.0436 0.2332 -0.19 0.8517   
pcs' -0.2952 1.0079 -0.29 0.7697   
pcs'' -3.1835 4.8079 -0.66 0.5082   
pcs''' 14.4216 8.3721 1.72 0.0857

## Plot all six fits?

p1 <- ggplot(Predict(mod\_B1)) + ggtitle("B1: Linear")  
p2 <- ggplot(Predict(mod\_B2)) + ggtitle("B2: Quadratic")  
p3 <- ggplot(Predict(mod\_B3)) + ggtitle("B3. Cubic")  
p4 <- ggplot(Predict(mod\_C3)) + ggtitle("C3: 3-knot RCS")  
p5 <- ggplot(Predict(mod\_C4)) + ggtitle("C4. 4-knot RCS")  
p6 <- ggplot(Predict(mod\_C5)) + ggtitle("C5. 5-knot RCS")  
  
(p1 + p2 + p3) / (p4 + p5 + p6)



## Which of these models looks better?

* I used set.seed(432) then validate(mod\_B1) etc.

| Model | Index-Corrected | Corrected MSE |
| --- | --- | --- |
| B1 (linear) | 0.0848 | 143.25 |
| B2 (quadratic) | 0.0752 | 142.49 |
| B3 (cubic) | 0.0909 | 143.73 |
| C3 (3-knot RCS) | 0.0732 | 143.31 |
| C4 (4-knot RCS) | 0.0870 | 144.00 |
| C5 (5-knot RCS) | 0.0984 | 141.44 |

* We’d need to look at residual plots, too.

## Next Time

* Determining how to spend degrees of freedom on non-linearity
* The HERS data
* Fitting a more complex linear regression model
* Adding missing data into all of this, and using multiple imputation

# Appendix: On Splines

## How complex should our spline be?

Restricted cubic splines can fit many different types of non-linearities. Specifying the number of knots is all you need to do in R to get a reasonable result from a restricted cubic spline.

The most common choices are 3, 4, or 5 knots.

* 3 Knots, 2 degrees of freedom, allows the curve to “bend” once.
* 4 Knots, 3 degrees of freedom, lets the curve “bend” twice.
* 5 Knots, 4 degrees of freedom, lets the curve “bend” three times.

## A simulated data set

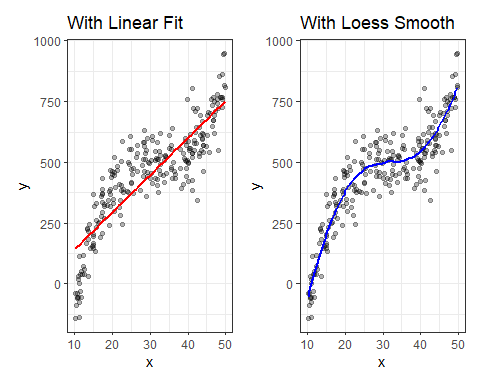
set.seed(4322024)  
  
sim\_data <- tibble(  
 x = runif(250, min = 10, max = 50),  
 y = 3\*(x-30) - 0.3\*(x-30)^2 + 0.05\*(x-30)^3 +   
 rnorm(250, mean = 500, sd = 70)  
)  
  
head(sim\_data)

# A tibble: 6 × 2  
 x y  
 <dbl> <dbl>  
1 31.9 581.   
2 10.4 -90.5  
3 38.0 476.   
4 15.6 232.   
5 39.6 474.   
6 42.5 721.

## The sim\_data, plotted.

p1 <- ggplot(sim\_data, aes(x = x, y = y)) +   
 geom\_point(alpha = 0.3) +  
 geom\_smooth(method = "lm", col = "red", se = FALSE) +  
 labs(title = "With Linear Fit")  
  
p2 <- ggplot(sim\_data, aes(x = x, y = y)) +   
 geom\_point(alpha = 0.3) +  
 geom\_smooth(method = "loess", col = "blue", se = FALSE) +  
 labs(title = "With Loess Smooth")  
  
p1 + p2

`geom\_smooth()` using formula = 'y ~ x'  
`geom\_smooth()` using formula = 'y ~ x'



## Fitting Non-Linear Terms with lm

We’ll fit:

* a linear model
* two models using orthogonal polynomials (poly()), and
* three models using restricted cubic splines (rcs())

sim\_linear <- lm(y ~ x, data = sim\_data)  
sim\_poly2 <- lm(y ~ poly(x, 2), data = sim\_data)  
sim\_poly3 <- lm(y ~ poly(x, 3), data = sim\_data)  
sim\_rcs3 <- lm(y ~ rcs(x, 3), data = sim\_data)  
sim\_rcs4 <- lm(y ~ rcs(x, 4), data = sim\_data)  
sim\_rcs5 <- lm(y ~ rcs(x, 5), data = sim\_data)

## augment() for fitted values and residuals

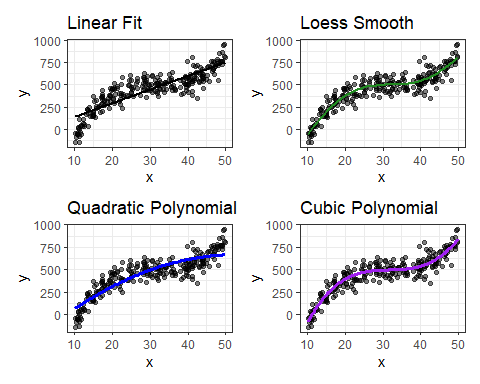
sim\_linear\_aug <- augment(sim\_linear, sim\_data)  
sim\_poly2\_aug <- augment(sim\_poly2, sim\_data)  
sim\_poly3\_aug <- augment(sim\_poly3, sim\_data)  
sim\_rcs3\_aug <- augment(sim\_rcs3, sim\_data)  
sim\_rcs4\_aug <- augment(sim\_rcs4, sim\_data)  
sim\_rcs5\_aug <- augment(sim\_rcs5, sim\_data)

This will help us to plot the fits for each of these six models.

## Add the Polynomial Fits

p1 <- ggplot(sim\_data, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_smooth(method = "lm", col = "black", se = F) +  
 labs(title = "Linear Fit")   
  
p2 <- ggplot(sim\_data, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_smooth(method = "loess", col = "forestgreen", se = F) +  
 labs(title = "Loess Smooth")   
  
p3 <- ggplot(sim\_poly2\_aug, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_line(aes(x = x, y = .fitted),   
 col = "blue", size = 1.25) +  
 labs(title = "Quadratic Polynomial")   
  
p4 <- ggplot(sim\_poly3\_aug, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_line(aes(x = x, y = .fitted),   
 col = "purple", size = 1.25) +  
 labs(title = "Cubic Polynomial")   
  
(p1 + p2) / (p3 + p4)

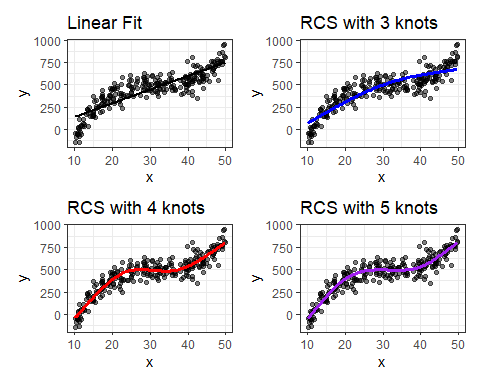
`geom\_smooth()` using formula = 'y ~ x'  
`geom\_smooth()` using formula = 'y ~ x'



## Restricted Cubic Spline Fits

p0 <- ggplot(sim\_data, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_smooth(method = "lm", col = "black", se = F) +  
 labs(title = "Linear Fit")   
  
p3 <- ggplot(sim\_rcs3\_aug, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_line(aes(x = x, y = .fitted),   
 col = "blue", size = 1.25) +  
 labs(title = "RCS with 3 knots")   
  
p4 <- ggplot(sim\_rcs4\_aug, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_line(aes(x = x, y = .fitted),   
 col = "red", size = 1.25) +  
 labs(title = "RCS with 4 knots")   
  
p5 <- ggplot(sim\_rcs5\_aug, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_line(aes(x = x, y = .fitted),   
 col = "purple", size = 1.25) +  
 labs(title = "RCS with 5 knots")   
  
(p0 + p3) / (p4 + p5)

`geom\_smooth()` using formula = 'y ~ x'



# Returning to the c4\_im data

## Load and Partition Data

This is from the c4im example we used last Thursday.

c4im <- read\_rds("c04/data/c4im.Rds")  
  
c4im <- c4im |>  
 mutate(fruit\_c = fruit\_day - mean(fruit\_day))  
  
set.seed(432) ## for future replication  
c4im\_split <- initial\_split(c4im, prop = 3/4)  
train\_c4im <- training(c4im\_split)  
test\_c4im <- testing(c4im\_split)

## Fitting Restricted Cubic Splines with lm and rcs

For most applications, three to five knots strike a nice balance between complicating the model needlessly and fitting data pleasingly. Let’s consider a restricted cubic spline model for 1000/bmi based on fruit\_c again, but now with:

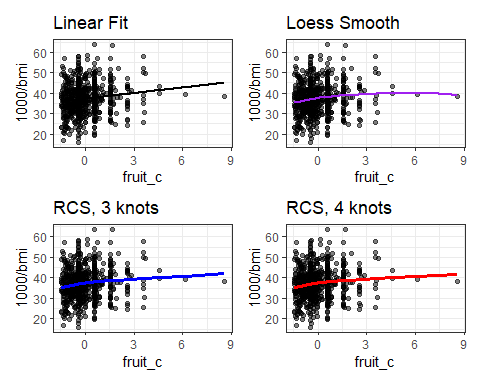
* in temp3, 3 knots, and
* in temp4, 4 knots,

temp3 <- lm(1000/bmi ~ rcs(fruit\_c, 3), data = train\_c4im)  
temp4 <- lm(1000/bmi ~ rcs(fruit\_c, 4), data = train\_c4im)

## Spline models for bmi and fruit\_c

temp3\_aug <- augment(temp3, train\_c4im)  
temp4\_aug <- augment(temp4, train\_c4im)  
  
p1 <- ggplot(train\_c4im, aes(x = fruit\_c, y = 1000/bmi)) +  
 geom\_point(alpha = 0.5) +  
 geom\_smooth(method = "lm", col = "black", se = F) +  
 labs(title = "Linear Fit")   
  
p2 <- ggplot(train\_c4im, aes(x = fruit\_c, y = 1000/bmi)) +  
 geom\_point(alpha = 0.5) +  
 geom\_smooth(method = "loess", col = "purple", se = F) +  
 labs(title = "Loess Smooth")   
  
p3 <- ggplot(temp3\_aug, aes(x = fruit\_c, y = 1000/bmi)) +  
 geom\_point(alpha = 0.5) +  
 geom\_line(aes(x = fruit\_c, y = .fitted),   
 col = "blue", size = 1.25) +  
 labs(title = "RCS, 3 knots")   
  
p4 <- ggplot(temp4\_aug, aes(x = fruit\_c, y = 1000/bmi)) +  
 geom\_point(alpha = 0.5) +  
 geom\_line(aes(x = fruit\_c, y = .fitted),   
 col = "red", size = 1.25) +  
 labs(title = "RCS, 4 knots")   
  
(p1 + p2) / (p3 + p4)

`geom\_smooth()` using formula = 'y ~ x'  
`geom\_smooth()` using formula = 'y ~ x'



## Let’s try an RCS with 4 knots

m\_4 <- lm(1000/bmi ~ rcs(fruit\_c, 4) + exerany + health,  
 data = train\_c4im)  
  
m\_4int <- lm(1000/bmi ~ rcs(fruit\_c, 4) + exerany \* health,  
 data = train\_c4im)

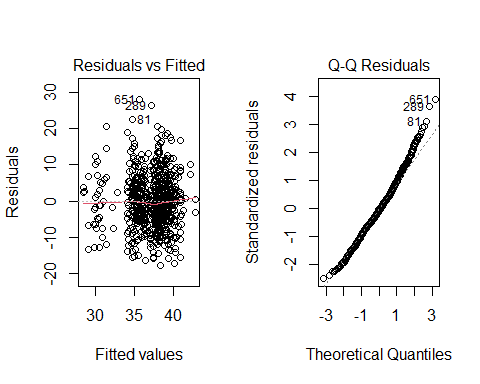
## m\_4int coefficients

tidy(m\_4int, conf.int = TRUE, conf.level = 0.90) |>  
 gt() |> fmt\_number(columns = estimate:conf.high, decimals = 3) |>  
 tab\_options(table.font.size = 18)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 36.968 | 2.284 | 16.186 | 0.000 | 33.206 | 40.730 |
| rcs(fruit\_c, 4)fruit\_c | 0.708 | 1.538 | 0.460 | 0.645 | -1.825 | 3.241 |
| rcs(fruit\_c, 4)fruit\_c' | 0.874 | 7.075 | 0.124 | 0.902 | -10.780 | 12.528 |
| rcs(fruit\_c, 4)fruit\_c'' | -3.021 | 20.152 | -0.150 | 0.881 | -36.216 | 30.173 |
| exerany | 2.574 | 1.888 | 1.363 | 0.173 | -0.536 | 5.684 |
| healthVG | 1.709 | 1.978 | 0.864 | 0.388 | -1.550 | 4.968 |
| healthG | -1.927 | 1.962 | -0.982 | 0.326 | -5.159 | 1.305 |
| healthF | -6.217 | 2.157 | -2.883 | 0.004 | -9.769 | -2.664 |
| healthP | -7.629 | 3.117 | -2.448 | 0.015 | -12.762 | -2.495 |
| exerany:healthVG | -2.955 | 2.183 | -1.353 | 0.176 | -6.551 | 0.641 |
| exerany:healthG | -1.929 | 2.187 | -0.882 | 0.378 | -5.530 | 1.673 |
| exerany:healthF | 4.878 | 2.512 | 1.942 | 0.053 | 0.740 | 9.017 |
| exerany:healthP | 5.802 | 3.673 | 1.579 | 0.115 | -0.249 | 11.853 |

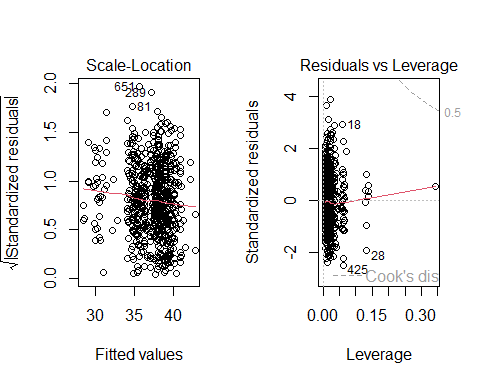
## m\_4int Residual Plots

par(mfrow = c(1,2)); plot(m\_4int, which = c(1,2))



## m\_4int Residual Plots

par(mfrow = c(1,2)); plot(m\_4int, which = c(3,5))



## How do models m\_4 and m\_4int do in testing?

m4\_test\_aug <- augment(m\_4, newdata = test\_c4im) |>  
 mutate(bmi\_fit = 1000/.fitted)  
m4int\_test\_aug <- augment(m\_4int, newdata = test\_c4im) |>  
 mutate(bmi\_fit = 1000/.fitted)  
  
testing\_r2 <- bind\_rows(  
 rsq(m4\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 rsq(m4int\_test\_aug, truth = bmi, estimate = bmi\_fit)) |>  
 mutate(model = c("m4", "m4int"))  
  
testing\_rmse <- bind\_rows(  
 rmse(m4\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 rmse(m4int\_test\_aug, truth = bmi, estimate = bmi\_fit)) |>  
 mutate(model = c("m4", "m4int"))  
  
testing\_mae <- bind\_rows(  
 mae(m4\_test\_aug, truth = bmi, estimate = bmi\_fit),  
 mae(m4int\_test\_aug, truth = bmi, estimate = bmi\_fit)) |>  
 mutate(model = c("m4", "m4int"))

## m\_4 and m\_4int in test sample

After back-transformation of fitted values of 1000/bmi to **bmi**:

bind\_cols(testing\_r2 |> select(model, rsquare = .estimate),   
 testing\_rmse |> select(rmse = .estimate),  
 testing\_mae |> select(mae = .estimate)) |>   
 gt() |> fmt\_number(columns = -model, decimals = 4) |>  
 tab\_options(table.font.size = 20)

| model | rsquare | rmse | mae |
| --- | --- | --- | --- |
| m4 | 0.0703 | 5.6461 | 4.3076 |
| m4int | 0.0326 | 5.8818 | 4.4816 |