432 Class 07

https://thomaselove.github.io/432-2025/

2025-02-04

## Today’s Agenda

* Splines and other non-linear terms
* Spearman’s plot: exploring non-linearity
  + Spending degrees of freedom wisely
* Linear Regression (HELP trial again)
  + A complex model with non-linear terms
  + Assessing fit with ols() and lm()
  + Calibration of the model
  + Prediction Intervals and Confidence Intervals

## Today’s R Setup

knitr::opts\_chunk$set(comment = NA)  
  
library(janitor)  
library(naniar)  
library(broom); library(gt); library(patchwork)  
library(haven)  
library(rms) ## auto-loads Hmisc  
library(easystats)  
library(tidyverse)  
  
theme\_set(theme\_bw())

## 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, lets the curve “bend” once.
* 4 Knots, 3 degrees of freedom, lets the curve “bend” twice.
* 5 Knots, 4 degrees of freedom; curve “bends” three times.

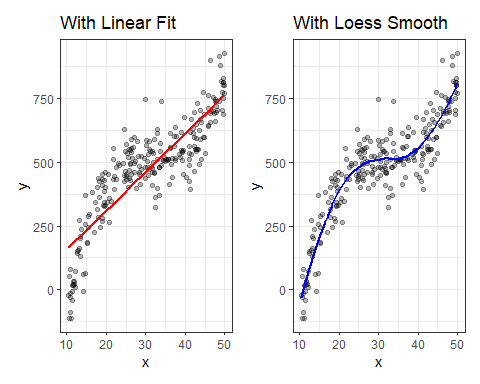
## A simulated data set

set.seed(20250204)  
  
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 19.6 430.  
2 43.7 666.  
3 33.5 368.  
4 43.5 493.  
5 29.7 515.  
6 45.0 535.

## The sim\_data, plotted.

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



## 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)

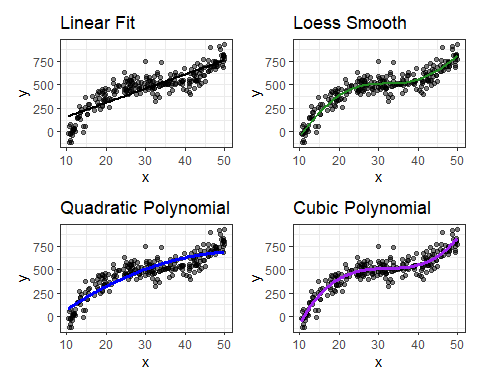
## augment() for our six models

This will generate fitted y predictions and residuals, which we can use to help us plot the fits for each of the six models we’ve generated using the simdata data.

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)

## Add the Polynomial Fits

p1 <- ggplot(sim\_data, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_smooth(method = "lm", formula = y ~ x,   
 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", formula = y ~ x,   
 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", linewidth = 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", linewidth = 1.25) +  
 labs(title = "Cubic Polynomial")   
  
(p1 + p2) / (p3 + p4)

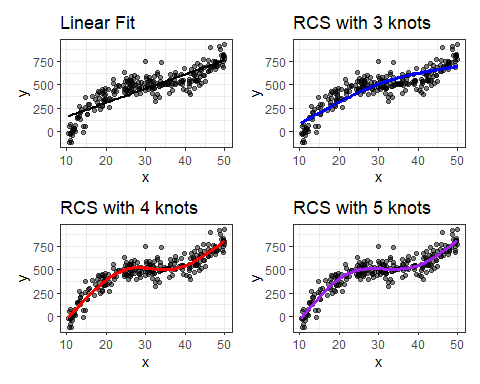


## Restricted Cubic Spline Fits

p0 <- ggplot(sim\_data, aes(x = x, y = y)) +  
 geom\_point(alpha = 0.5) +  
 geom\_smooth(method = "lm", formula = y ~ x,   
 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")

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

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)



# Deciding Where to Try Non-Linear Terms

## Spending degrees of freedom wisely

* Suppose we have many possible predictors, and minimal theory or subject matter knowledge to guide us.
* We might want our final inferences to be as unbiased as possible. To accomplish this, we have to pay a penalty (in terms of degrees of freedom) for any “peeks” we make at the data in advance of fitting a model.
* So that rules out a lot of decision-making about non-linearity based on looking at the data, if our sample size isn’t incredibly large.

## Back to the HELP Trial

Health Evaluation and Linkage to Primary Care (HELP) was a clinical trial of adult inpatients recruited from a detoxification unit.

* We have baseline data for each subject on several variables, including two outcomes:

| Variable | Description |
| --- | --- |
| cesd | Center for Epidemiologic Studies-Depression |
| cesd\_hi | cesd above 15 (indicates high risk) |

## help1 data load

help1 <- tibble(mosaicData::HELPrct) |>  
 select(id, cesd, age, sex, subst = substance, mcs, pcs, pss\_fr) |>  
 zap\_label() |>  
 mutate(across(where(is.character), as\_factor),   
 id = as.character(id),   
 cesd\_hi = factor(as.numeric(cesd >= 16)))  
  
dim(help1); n\_miss(help1)

[1] 453 9

[1] 0

head(help1, 5)

# A tibble: 5 × 9  
 id cesd age sex subst mcs pcs pss\_fr cesd\_hi  
 <chr> <int> <int> <fct> <fct> <dbl> <dbl> <int> <fct>   
1 1 49 37 male cocaine 25.1 58.4 0 1   
2 2 30 37 male alcohol 26.7 36.0 1 1   
3 3 39 26 male heroin 6.76 74.8 13 1   
4 4 15 39 female heroin 44.0 61.9 11 0   
5 5 39 32 male cocaine 21.7 37.3 10 1

## The Six Predictors in help1

* Predict cesd using these six predictors…

| Variable | Description |
| --- | --- |
| age | subject age (in years) |
| sex | female (n = 107) or male (n = 346) |
| subst | substance abused (alcohol, cocaine, heroin) |
| mcs | SF-36 Mental Component Score |
| pcs | SF-36 Physical Component Score |
| pss\_fr | perceived social support by friends |

## Adding Non-Linear Terms Spends DF

What happens when we add a non-linear term?

* A polynomial of degree D costs D degrees of freedom.
  + So a polynomial of degree 2 (quadratic) costs 2 df, or 1 more than the main effect alone.
* A restricted cubic spline with K knots costs K-1 df.
  + So adding a spline with 4 knots uses 3 df, or 2 more than the main effect alone.
  + We’ll only consider splines with 3, 4, or 5 knots.

## Adding Non-Linear Terms Spends DF

Adding an interaction (product term) depends on the main effects of the predictors we are interacting

* If the product term’s predictors have df1 and df2 degrees of freedom, product term adds df1 x df2 degrees of freedom.
  + An interaction of a binary and quantitative variable adds 1 x 1 = 1 more df to the main effects model.
* When we use a quantitative variable in a spline and interaction, we’ll do the interaction on the main effect, not the spline.

## Spearman’s plot: A smart first step?

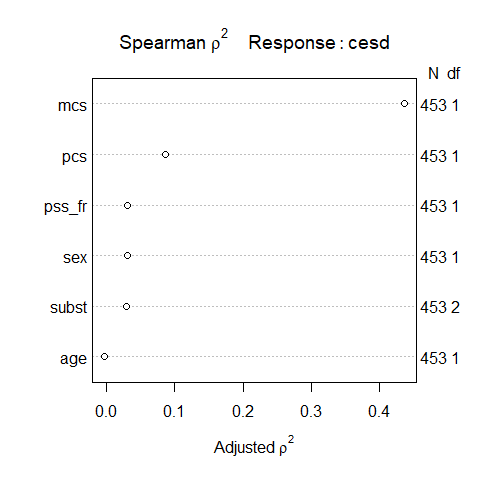
Spearman’s is an indicator (not a perfect one) of potential predictive punch, but doesn’t give away the game.

* Looking at Spearman’s and selecting predictors to include non-linearity for reduces the impact of “looking at the data” which leads to bias in the model.
* Idea: Perhaps we should focus our efforts re: non-linearity on predictors that score better on this measure.

spear\_cesd <- spearman2(cesd ~ mcs + subst + pcs + age + sex + pss\_fr,   
 data = help1)

## Spearman’s Plot

plot(spear\_cesd)



## Conclusions from Spearman Plot

* mcs is the most attractive candidate for a non-linear term, as it packs the most potential predictive punch, so if it does turn out to need non-linear terms, our degrees of freedom will be well spent.
  + This **does not** mean that mcs actually needs a non-linear term, or will show meaningfully better results if a non-linear term is included. We’d have to fit a model with and without non-linearity in mcs to know that.

## Conclusions from Spearman Plot

* pcs, also quantitative, has the next most potential predictive punch after mcs.
* pss\_fr and sex follow, then subst and age.

spear\_cesd

Spearman rho^2 Response variable:cesd  
  
 rho2 F df1 df2 P Adjusted rho2 n  
mcs 0.438 350.89 1 451 0.0000 0.436 453  
subst 0.034 7.97 2 450 0.0004 0.030 453  
pcs 0.089 44.22 1 451 0.0000 0.087 453  
age 0.000 0.12 1 451 0.7286 -0.002 453  
sex 0.033 15.56 1 451 0.0001 0.031 453  
pss\_fr 0.033 15.57 1 451 0.0001 0.031 453

## A Main Effects Model

Here’s a summary of the degrees of freedom for a main effects model without any non-linear terms.

fit1 <- lm(cesd ~ mcs + subst + pcs + age + sex + pss\_fr, data = help1)  
  
glance(fit1) |> select(df, df.residual, nobs) |>   
 gt() |> tab\_options(table.font.size = 20) |>   
 opt\_stylize(style = 3, color = "cyan")

| df | df.residual | nobs |
| --- | --- | --- |
| 7 | 445 | 453 |

We started with 453 observations (452 df) and fitting fit1 leaves 445 residual df, so fit1 uses 7 degrees of freedom.

## Grim Reality

One popular standard for linear regression requires at least 25 observations *per regression coefficient that you will estimate*[[1]](#footnote-53).

* With 453 observations (452 df) in the HELP trial, we should be thinking about models with modest numbers of regression inputs, since 25 is really a bare minimum.
* We’ve already committed to 7 such coefficients (intercept + our six predictors.)

## Sample Size (spending df)

* Non-linear terms (polynomials, splines, product terms) just add to the problem, as they need additional degrees of freedom (parameters) to be estimated.
* We’ll also use more df every time if we consider re-fitting after variable selection.

So we might choose to include non-linear terms in just two or three variables with this modest sample size (n = 453).

* But I’ll ignore all of that (for now) and propose a complex fit2 model …

## Proposed New Model fit2

Fit a model to predict cesd using:

* a 5-knot spline on mcs
* a 3-knot spline on pcs
* a linear term on pss\_fr
* a linear term on age
* an interaction of sex with the main effect of mcs (restricting our model so that terms that are non-linear in both sex and mcs are excluded), and
* a main effect of subst

## Our new model fit2

Definitely more than we can reasonably do with 453 observations, but let’s see how it looks.

dd <- datadist(help1)  
options(datadist = "dd")  
  
fit2 <- ols(cesd ~ rcs(mcs, 5) + rcs(pcs, 3) + sex + mcs %ia% sex +   
 pss\_fr + age + subst,   
 data = help1, x = TRUE, y = TRUE)

* %ia% tells R to fit an interaction term with sex and the main effect of mcs.
  + We have to include sex as a main effect for the interaction term (%ia%) to work. We already have the main effect of mcs in as part of the spline.

## Can we fit2 with lm()?

Yes.

fit2\_lm <- lm(cesd ~ rcs(mcs, 5) + rcs(pcs, 3) + sex + mcs %ia% sex +   
 pss\_fr + age + subst, data = help1)  
  
glance(fit2\_lm) |> select(df, df.residual, nobs) |>   
 gt() |> tab\_options(table.font.size = 20) |>   
 opt\_stylize(style = 3, color = "cyan")

| df | df.residual | nobs |
| --- | --- | --- |
| 12 | 440 | 453 |

* So fit2\_lm uses an additional 5 degrees of freedom beyond the 7 in fit1.

## Our fitted model fit2 (from ols())

fit2

Linear Regression Model  
  
ols(formula = cesd ~ rcs(mcs, 5) + rcs(pcs, 3) + sex + mcs %ia%   
 sex + pss\_fr + age + subst, data = help1, x = TRUE, y = TRUE)  
  
 Model Likelihood Discrimination   
 Ratio Test Indexes   
Obs 453 LR chi2 349.44 R2 0.538   
sigma8.6248 d.f. 12 R2 adj 0.525   
d.f. 440 Pr(> chi2) 0.0000 g 10.439   
  
Residuals  
  
 Min 1Q Median 3Q Max   
-26.7893 -5.9000 0.1545 5.5884 26.1304   
  
 Coef S.E. t Pr(>|t|)  
Intercept 76.3346 6.2540 12.21 <0.0001   
mcs -0.9306 0.2315 -4.02 <0.0001   
mcs' 1.6607 2.5040 0.66 0.5075   
mcs'' -2.8854 8.3945 -0.34 0.7312   
mcs''' 0.2942 7.9390 0.04 0.9705   
pcs -0.2341 0.0883 -2.65 0.0083   
pcs' -0.0151 0.1000 -0.15 0.8797   
sex=male -2.0330 2.5456 -0.80 0.4249   
mcs \* sex=male -0.0129 0.0783 -0.17 0.8690   
pss\_fr -0.2569 0.1046 -2.46 0.0144   
age -0.0466 0.0569 -0.82 0.4139   
subst=cocaine -2.6999 0.9965 -2.71 0.0070   
subst=heroin -2.1741 1.0677 -2.04 0.0423

## ANOVA for fit2

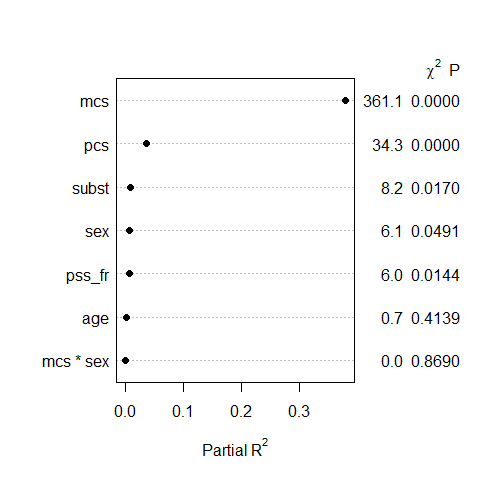
This ANOVA testing is sequential, other than the TOTALS.

anova(fit2)

Analysis of Variance Response: cesd   
  
 Factor d.f. Partial SS MS F P   
 mcs (Factor+Higher Order Factors) 5 26857.364671 5371.472934 72.21 <.0001  
 All Interactions 1 2.026255 2.026255 0.03 0.8690  
 Nonlinear 3 293.502251 97.834084 1.32 0.2688  
 pcs 2 2548.388579 1274.194290 17.13 <.0001  
 Nonlinear 1 1.705031 1.705031 0.02 0.8797  
 sex (Factor+Higher Order Factors) 2 451.578352 225.789176 3.04 0.0491  
 All Interactions 1 2.026255 2.026255 0.03 0.8690  
 mcs \* sex (Factor+Higher Order Factors) 1 2.026255 2.026255 0.03 0.8690  
 pss\_fr 1 448.812293 448.812293 6.03 0.0144  
 age 1 49.758786 49.758786 0.67 0.4139  
 subst 2 611.625952 305.812976 4.11 0.0170  
 TOTAL NONLINEAR 4 293.512204 73.378051 0.99 0.4146  
 TOTAL NONLINEAR + INTERACTION 5 294.601803 58.920361 0.79 0.5558  
 REGRESSION 12 38058.315322 3171.526277 42.64 <.0001  
 ERROR 440 32730.174744 74.386761

## Plotting ANOVA results for fit2

plot(anova(fit2), what = "partial R2", sort = "ascending")



## Validation of Summary Statistics

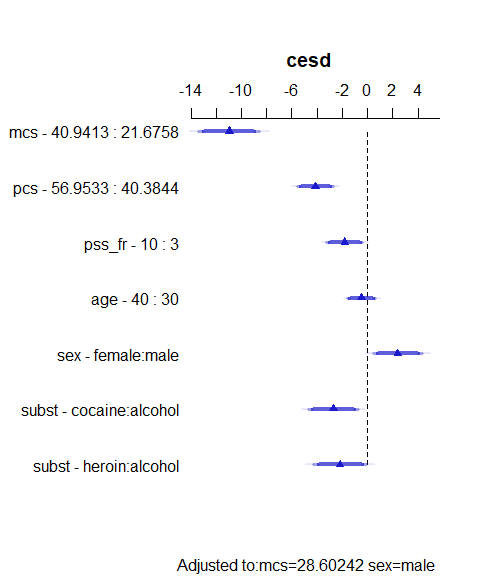
set.seed(432); validate(fit2, method = "boot", B = 300)

index.orig training test optimism index.corrected n  
R-square 0.5376 0.5523 0.5223 0.0300 0.5076 300  
MSE 72.2520 69.5508 74.6484 -5.0976 77.3497 300  
g 10.4392 10.5319 10.2926 0.2393 10.1998 300  
Intercept 0.0000 0.0000 0.7877 -0.7877 0.7877 300  
Slope 1.0000 1.0000 0.9755 0.0245 0.9755 300

* I’m making a blanket recommendation that you run 300 bootstrap validations unless I’ve told you specifically to do something else.

## summary results for fit2

plot(summary(fit2))



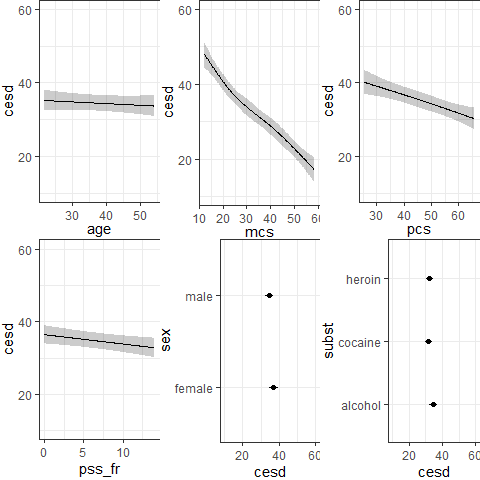
## summary results for fit2

summary(fit2)

Effects Response : cesd   
  
 Factor Low High Diff. Effect S.E. Lower 0.95 Upper 0.95  
 mcs 21.676 40.941 19.266 -10.96400 1.23340 -13.38800 -8.539800   
 pcs 40.384 56.953 16.569 -4.10790 0.73381 -5.55010 -2.665700   
 pss\_fr 3.000 10.000 7.000 -1.79860 0.73225 -3.23780 -0.359500   
 age 30.000 40.000 10.000 -0.46552 0.56918 -1.58420 0.653130   
 sex - female:male 2.000 1.000 NA 2.40260 0.99054 0.45577 4.349300   
 subst - cocaine:alcohol 1.000 2.000 NA -2.69990 0.99647 -4.65830 -0.741430   
 subst - heroin:alcohol 1.000 3.000 NA -2.17410 1.06770 -4.27250 -0.075632   
  
Adjusted to: mcs=28.60242 sex=male

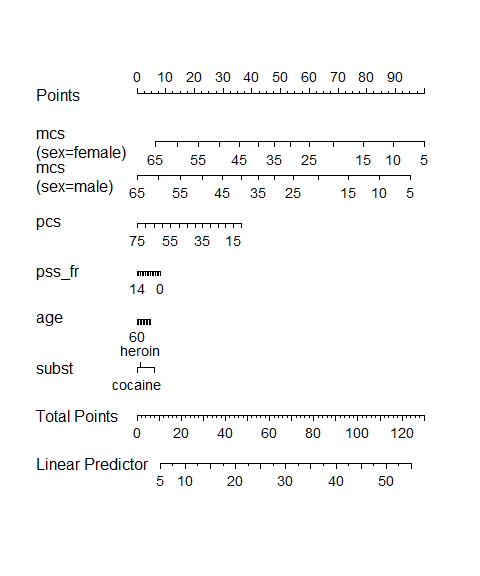
## Impact of non-linearity?

ggplot(Predict(fit2))



## Nomogram for fit2

plot(nomogram(fit2))



## How to use the nomogram

1. 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”.
2. Then sum up the points over all predictors to obtain “Total Points”.
3. Draw a vertical line from “Total Points” to “Linear Predictor” to obtain predicted cesd.

The nomogram shows modeled effects and their impact on the predicted outcome.

## Making Predictions

Suppose we want to use our model fit2 to make a prediction for cesd for a new subject, named Grace, who has the following characteristics…

* sex = female, mcs = 40, pcs = 50
* pss\_fr = 7, age = 45, subst = “cocaine”

We can build point and interval estimates for predicted cesd from fit2 as follows…

## Predictions for an Individual

Suppose we have a new **individual subject** named Grace.

grace <- tibble(sex = "female", mcs = 40, pcs = 50,   
 pss\_fr = 7, age = 45, subst = "cocaine")  
  
predict(fit2, newdata = grace, conf.int = 0.95, conf.type = "individual") |>  
 as\_vector()

linear.predictors.1 lower.1 upper.1   
 27.88915 10.64701 45.13129

Our predicted cesd for Grace is 27.89, with 95% **prediction interval** (10.65, 45.13).

## Predictions for a Long-Run Mean

Predict **mean** cesd of a set of subjects with Grace’s predictor values, along with a **confidence interval**.

predict(fit2, newdata = grace, conf.int = 0.95, conf.type = "mean") |>  
 as\_vector()

linear.predictors.1 lower.1 upper.1   
 27.88915 24.73335 31.04496

* Confidence interval (24.73, 31.04) is much narrower than prediction interval (10.65, 45.13).

## Assessing the Calibration of fit2

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

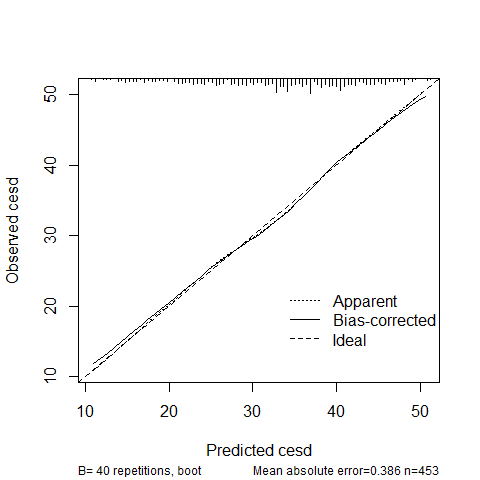
* Suppose our model assigns a predicted outcome of 6 to several subjects.
* If the model is well-calibrated, this means we expect the mean of those subjects’ actual outcomes to be very close to 6.
* We’d like to look at the relationship between the observed cesd outcome and our predicted cesd from the model.

## Building a Calibration Plot

* 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 a fit with ols().

## Checking the model’s calibration

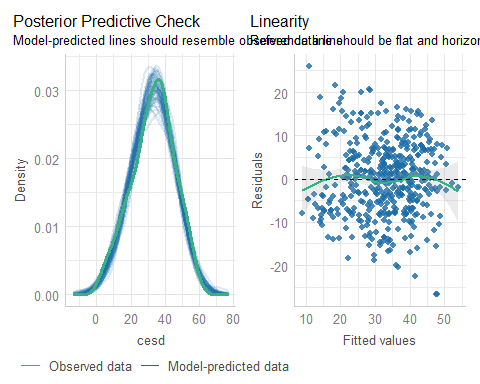
set.seed(432); plot(calibrate(fit2))



n=453 Mean absolute error=0.386 Mean squared error=0.19775  
0.9 Quantile of absolute error=0.704

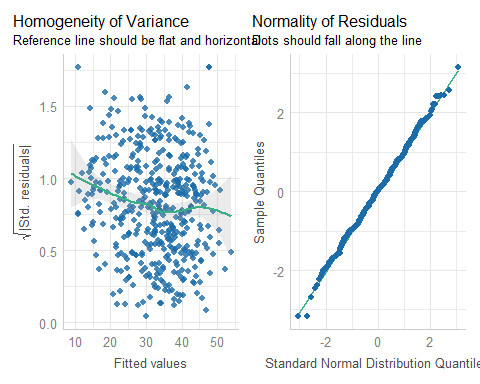
## Checking the Model (slide 1/3)

check\_model(fit2\_lm, check = c("pp\_check", "linearity"))



## Checking the Model (slide 2/3)

check\_model(fit2\_lm, detrend = FALSE, check = c("homogeneity", "qq"))



## Checking the Model (slide 3/3)

The collinearity plot is a bit hard to see with all of these terms, so we can just look at the variance inflation factors:

rms::vif(fit2)

mcs mcs' mcs'' mcs''' pcs   
 53.689536 4838.325134 12475.879348 2489.141282 5.505056   
 pcs' sex=male mcs \* sex=male pss\_fr age   
 5.364435 7.119400 11.829120 1.061218 1.170267   
 subst=cocaine subst=heroin   
 1.348167 1.380161

car::vif(fit2\_lm)

GVIF Df GVIF^(1/(2\*Df))  
rcs(mcs, 5) 5.443532 4 1.235905  
rcs(pcs, 3) 1.298865 2 1.067557  
sex 7.119400 1 2.668220  
mcs %ia% sex 11.829120 1 3.439349  
pss\_fr 1.061218 1 1.030155  
age 1.170267 1 1.081789  
subst 1.214244 2 1.049727

## Using both lm() and ols()

* We can and will regularly use both lm and ols to fit a model like fit2.

To delve into the details of how well this complex model works, and to help plot what is actually being fit, we’ll want to fit the model using ols().

* In Project A, we expect some results that are most easily obtained using lm() and others that are most easily obtained using ols().

## Next Example: Class 08 (2025-02-06)

* Focus on logistic regression with a new data set
  + Thinking about various pseudo- approaches
  + Developing an optimal cutpoint for a confusion matrix
  + Brier scores and other measures of calibration in logistic regression
  + Checking assumptions in logistic regression
  + Just about everything we might want to do…

1. Jenkins and Quintana-Ascensio [A solution to minimum sample size for regressions](https://pubmed.ncbi.nlm.nih.gov/32084211/). *PLoS One*, 2020 Feb 21; 15(2): e0229345. [↑](#footnote-ref-53)