432 Class 06

Thomas E. Love, Ph.D.

2026-01-29

Today's Agenda

- Splines and other non-linear terms
- Spearman's ρ^2 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

```r
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-linearity. Specifying the number of knots (usually 3, 4 or 5) is all you need to define.

- 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(20260129)

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  24.4  421.
2  46.9  582.
3  38.6  492.
4  29.1  443.
5  16.4  258.
6  29.2  505.
```

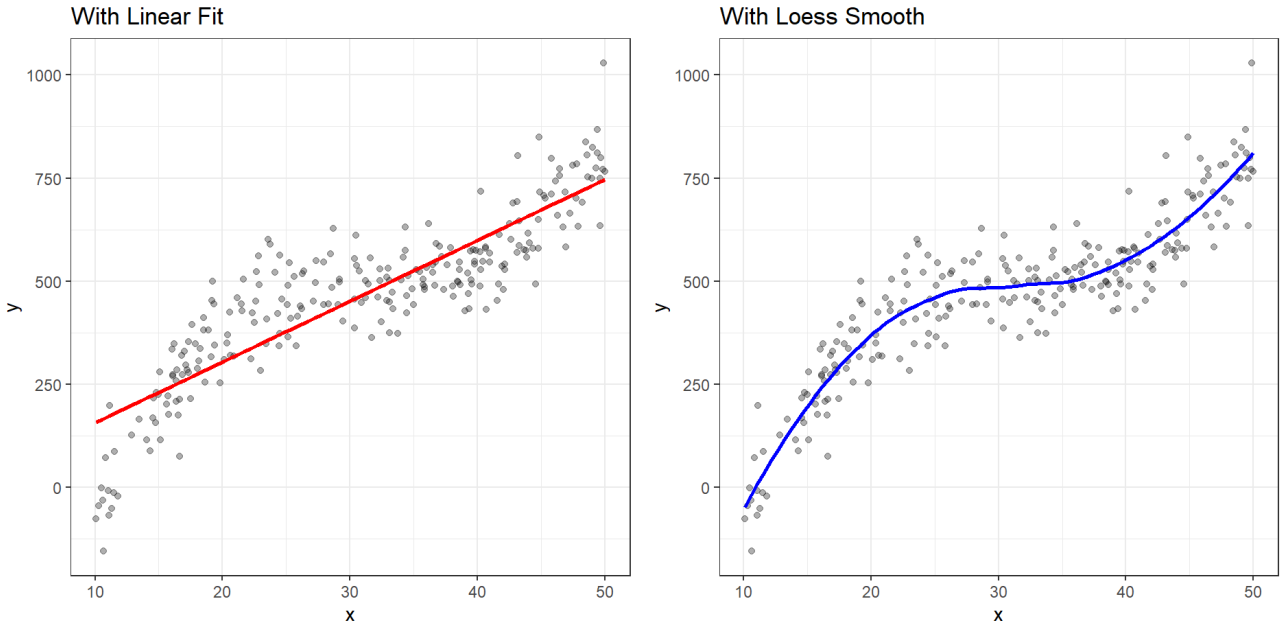
The `sim_data`, plotted.

```
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
```


The `sim_data`, plotted.



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)
```


Degrees of Freedom for each model

- We can check df with `anova(modelname)`

Formula	Model df	Resid. df	# obs.
`lm(y ~ x)`	1	248	250
`lm(y ~ poly(x,2))`	2	247	250
`lm(y ~ poly(x,3))`	3	246	250
`lm(y ~ rcs(x,3))`	2	247	250
`lm(y ~ rcs(x,4))`	3	246	250
`lm(y ~ rcs(x,5))`	4	245	250

`augment()` for our six models

`augment()` generates fitted `y` predictions and residuals, which will help us plot the fits for our six models.

```
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)

sim_linear_aug |> slice(1:2) |>
  gt() |> fmt_number(decimals = 3) |> tab_options(table.font.size = 20)
```

x	y	.fitted	.resid	.hat	.sigma	.cooksd	.std.resid
24.351	421.499	368.381	53.118	0.005	96.223	0.001	0.554
46.934	582.366	701.362	−118.996	0.012	95.981	0.009	−1.246

Add the Polynomial Fits

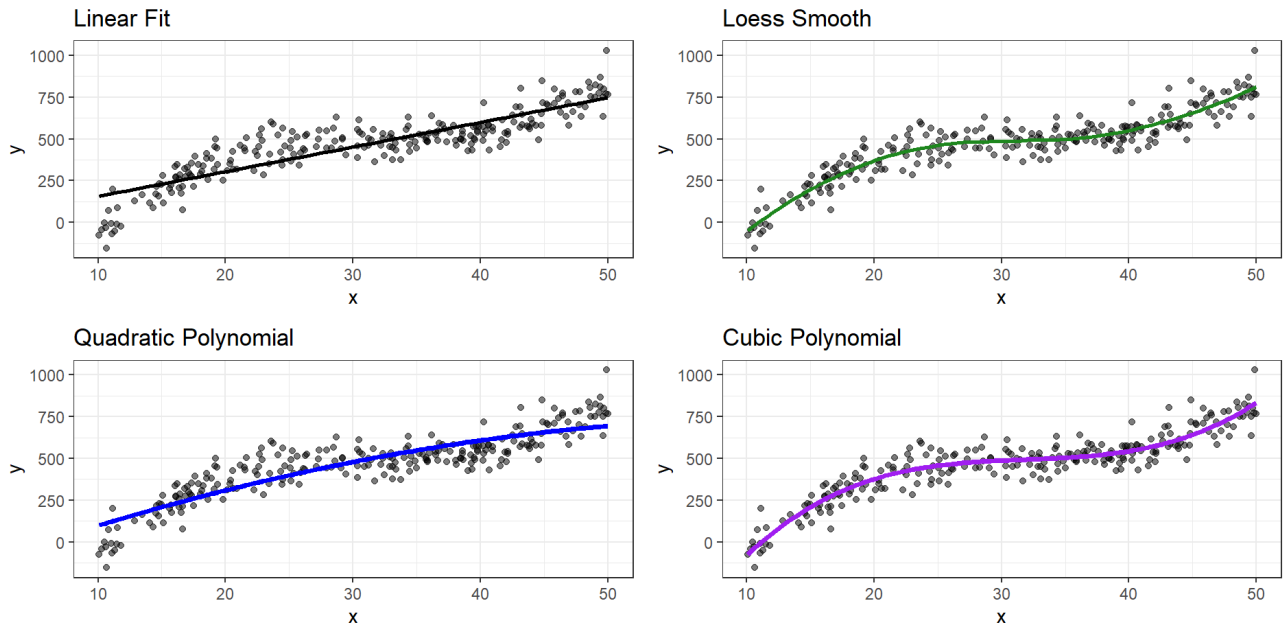
```
p1 <- ggplot(sim_data, aes(x = x, y = y)) +
    geom_point(alpha = 0.5) +
    geom_smooth(method = "lm", formula = y ~ x,
                col = "black", se = F) +
    labs(title = "Linear Fit")

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")

```

Add the Polynomial Fits



Restricted Cubic Spline Fits

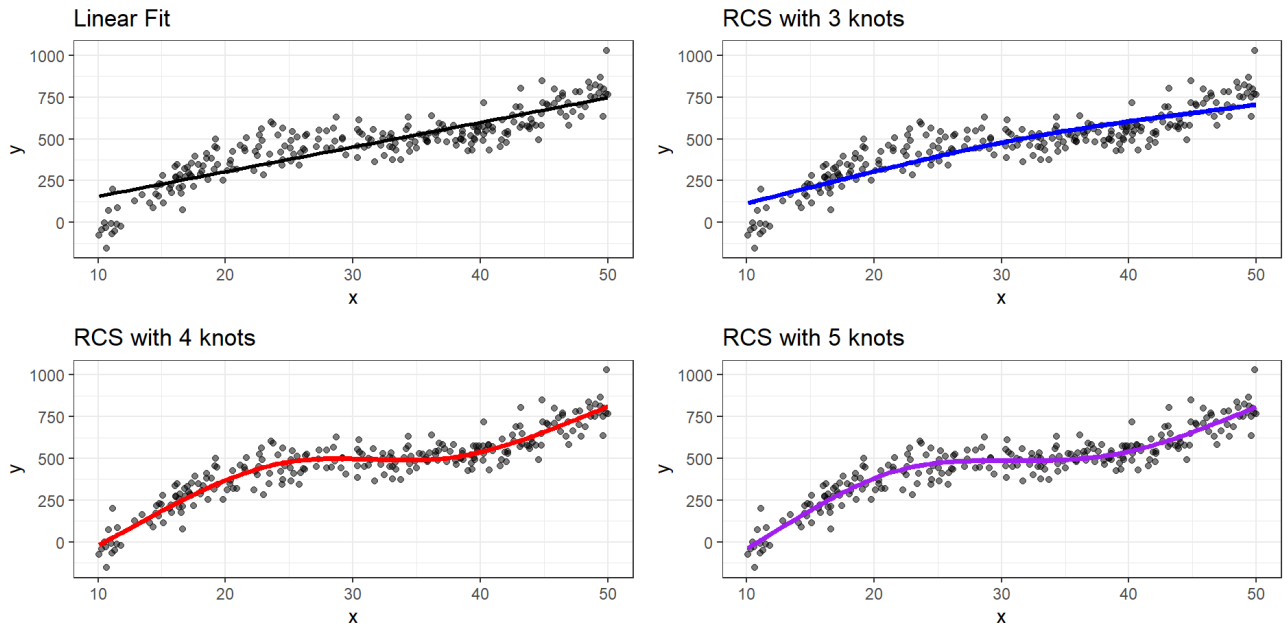
```
p0 <- ggplot(sim_data, aes(x = x, y = y)) +
    geom_point(alpha = 0.5) +
    geom_smooth(method = "lm", formula = y ~ x,
                col = "black", se = F) +
    labs(title = "Linear Fit")

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")

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")

```

Restricted Cubic Spline Fits



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

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.

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.

A smart first step?

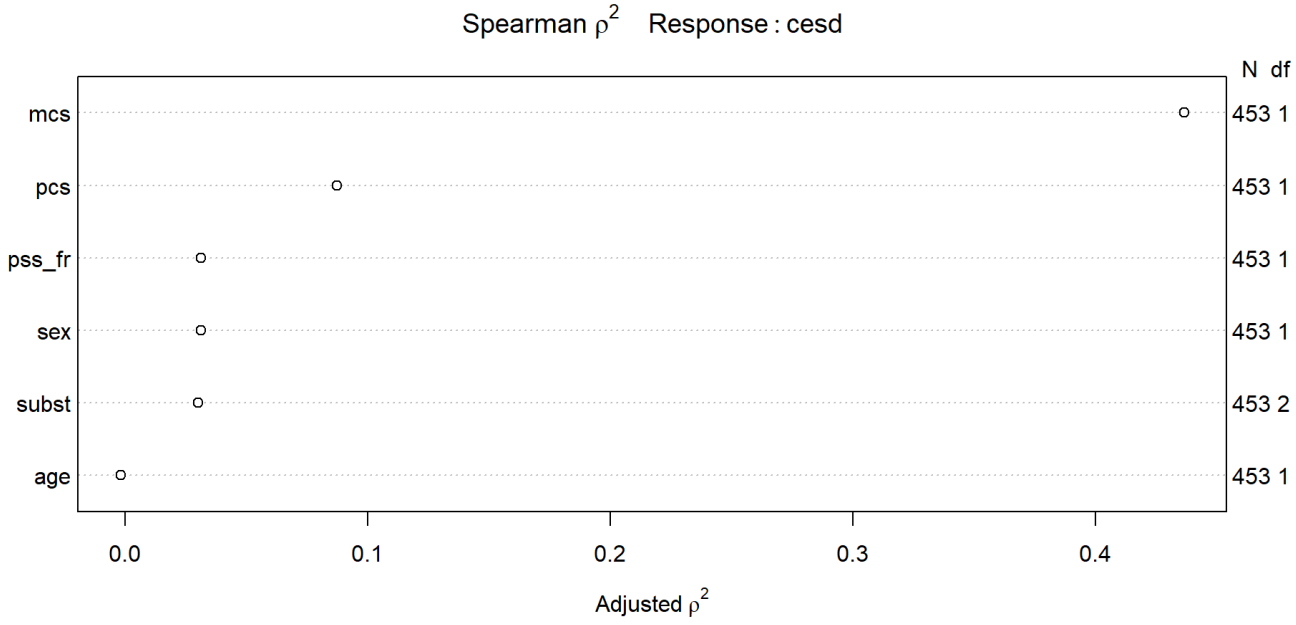
Spearman's is an indicator (not a perfect one) of potential predictive punch, but doesn't give away the game.

- 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`.

```
1 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].

- 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 polynomial of degree 2 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`

Standing Break

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 +
              pol(pss_fr,2) + 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. Note `poly()` in our `lm()` fit, rather than `pol()`.

```
fit2_lm <- lm(cesd ~ rcs(mcs, 5) + rcs(pcs, 3) + sex + mcs %ia% sex +
                poly(pss_fr,2) + 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
13	439	453

- So `fit2_lm` uses an additional 6 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 + pol(pss_fr, 2) + age + subst, data = help1, x = TRUE,
    y = TRUE)

                Model Likelihood    Discrimination
                      Ratio Test           Indexes
Obs     453    LR chi2    353.70    R2       0.542
sigma8.5942    d.f.           13    R2 adj   0.528
d.f.    439    Pr(> chi2) 0.0000    g       10.483

Residuals
```

ANOVA for `fit2`

This ANOVA testing is sequential, other than the TOTALS.

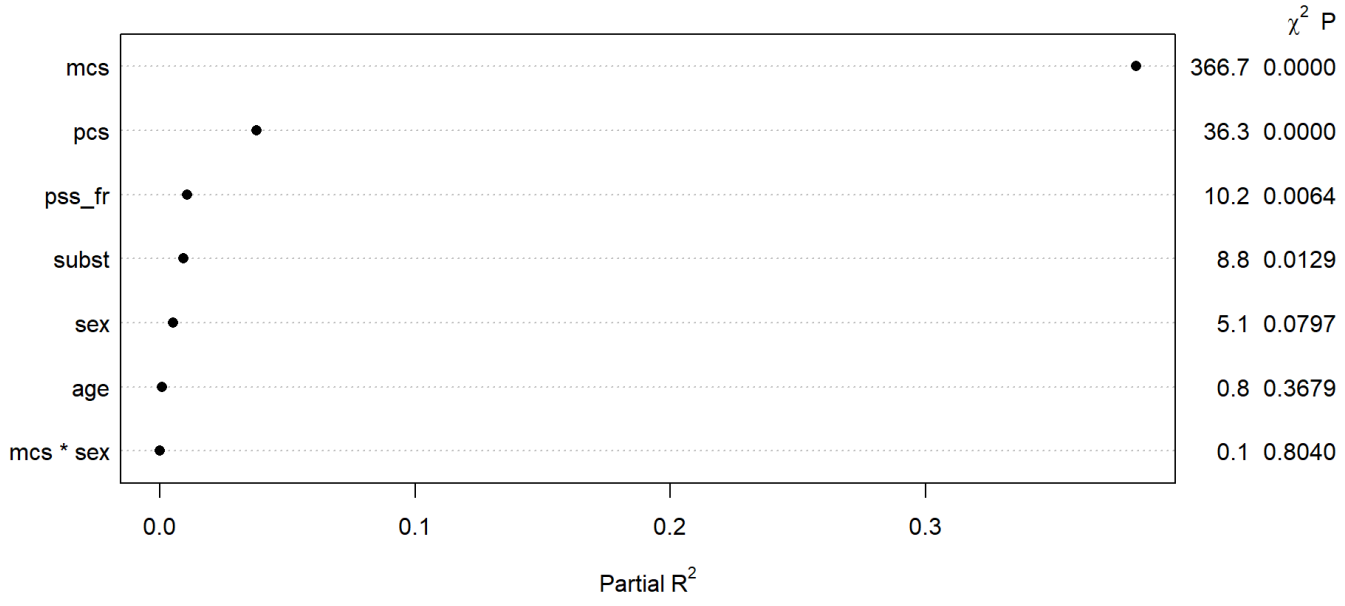
```
1 anova(fit2)
```

```
                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
```


Plotting ANOVA results for `fit2`

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



Validation of Summary Statistics

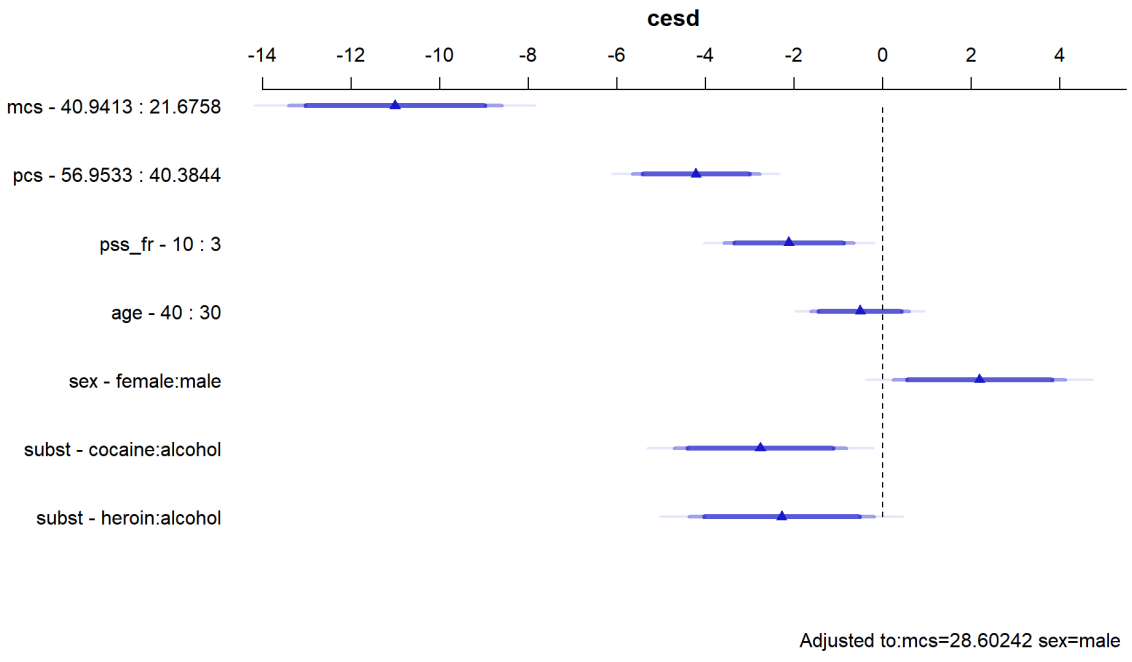
```
1 set.seed(432); validate(fit2, method = "boot", B = 300)
```

```
          index.orig training    test optimism index.corrected   Lower   Upper
R-square      0.5420   0.5570  0.5259   0.0311          0.5108  0.4445  0.5735
MSE          71.5769  68.8206 74.0829  -5.2623         76.8393 68.1247 85.9959
g            10.4826  10.5784 10.3304   0.2480         10.2346  9.1147 11.2645
Intercept     0.0000   0.0000  0.7935  -0.7935          0.7935 -3.2042  4.4715
Slope         1.0000   1.0000  0.9753   0.0247          0.9753  0.8738  1.0878
              n
R-square    300
MSE         300
g           300
Intercept   300
Slope       300
```

- I'm making a blanket recommendation that you run 300 bootstrap validations unless (in a Lab or something) I've told you specifically to do something else.

`summary` results for `fit2`

```
1 plot(summary(fit2))
```



`summary` results for `fit2`

```
1 summary(fit2)
```

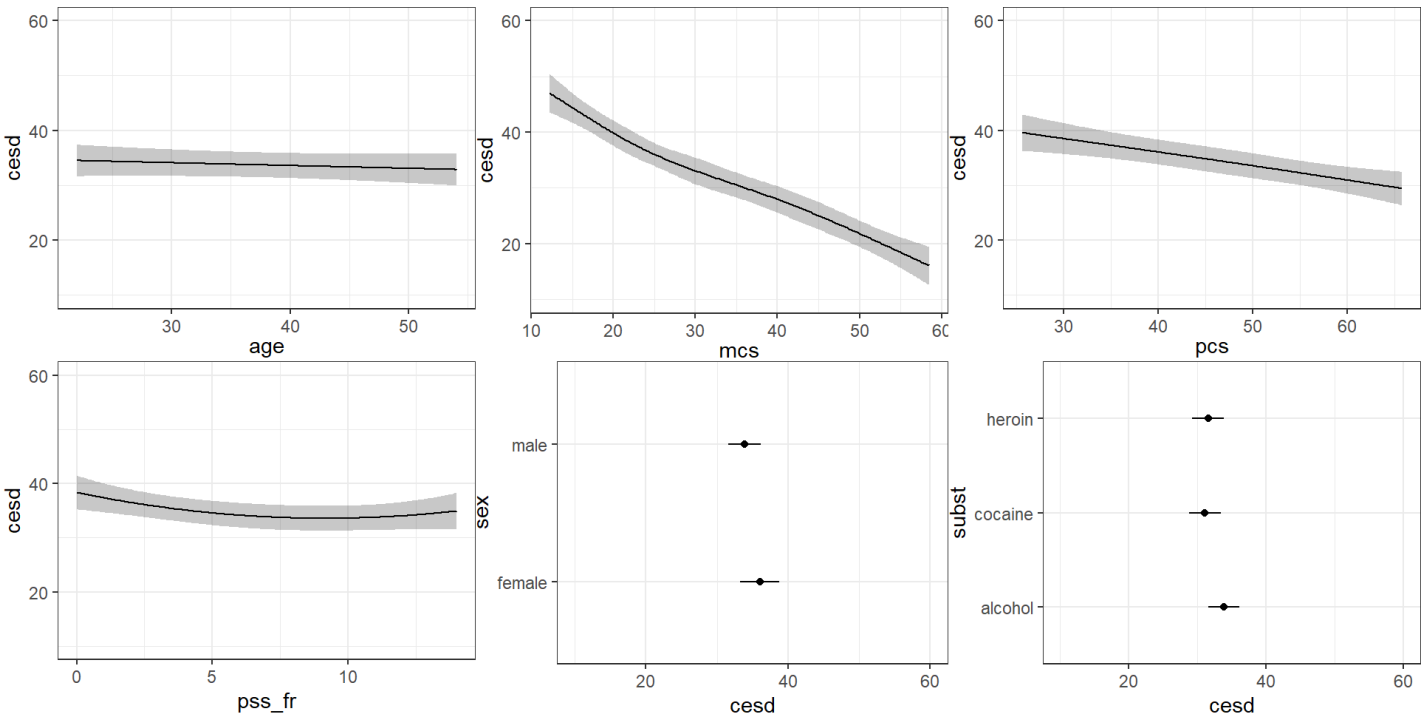
```
            Effects              Response : cesd

 Factor                    Low    High   Diff.  Effect    S.E.    Lower 0.95 Upper 0.95
 mcs                       21.676 40.941 19.266 -11.01300 1.22920 -13.42900  -8.59710
 pcs                       40.384 56.953 16.569  -4.21690 0.73316  -5.65780  -2.77590
 pss_fr                     3.000 10.000  7.000  -2.12120 0.74667  -3.58870  -0.65369
 age                       30.000 40.000 10.000  -0.51164 0.56762  -1.62720   0.60394
 sex - female:male          2.000  1.000     NA   2.18360 0.99288   0.23218   4.13500
 subst - cocaine:alcohol    1.000  2.000     NA  -2.76380 0.99343  -4.71630  -0.81134
 subst - heroin:alcohol     1.000  3.000     NA  -2.28280 1.06530  -4.37640  -0.18915

Adjusted to: mcs=28.60242 sex=male
```

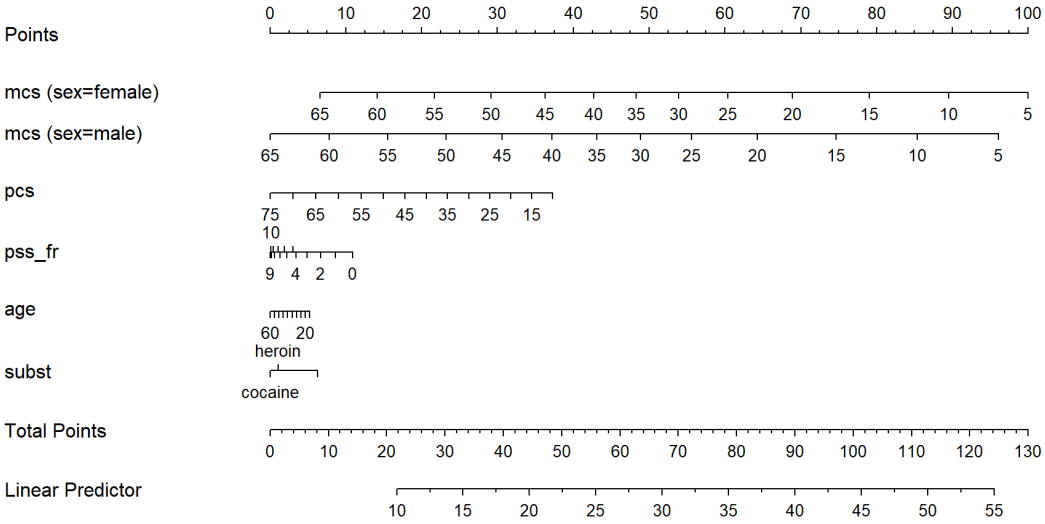

Impact of non-linearity?

```
1 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
          26.808537            9.595825           44.021249
```

Our predicted `cesd` for Grace is 26.81, with 95% **prediction interval** (9.60, 44.02).

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
           26.80854            23.49523            30.12185
```

- Confidence interval (23.50, 30.12) is much narrower than prediction interval (9.60, 44.02).

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.

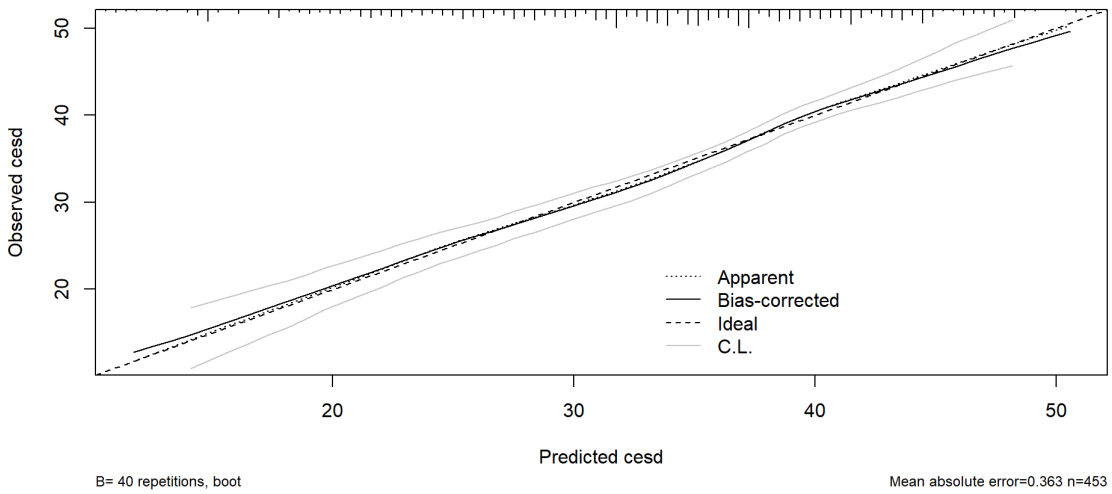
Building a Calibration Plot

- 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

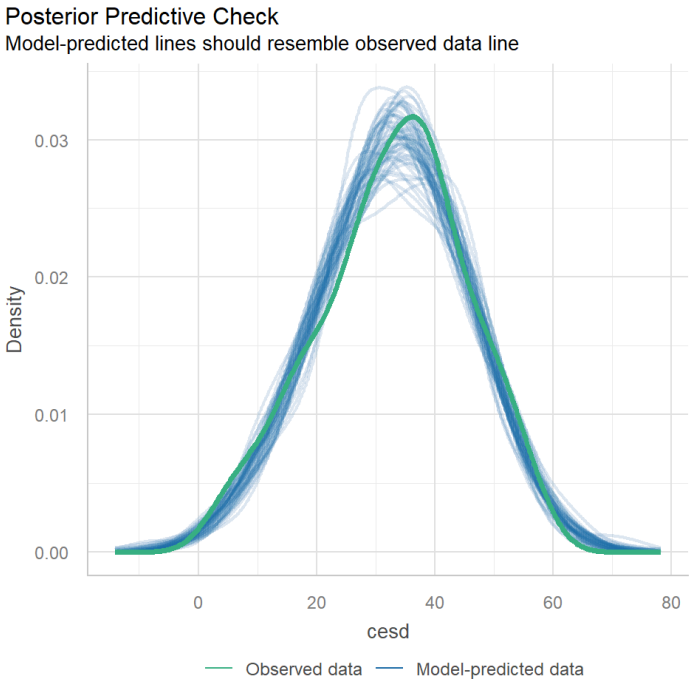
```
1 set.seed(432); plot(calibrate(fit2))
```

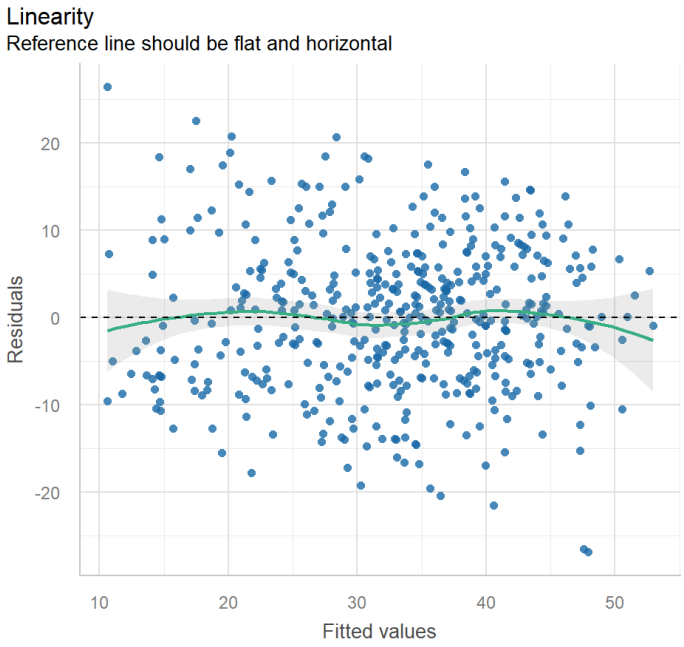
```
n=453   Mean absolute error=0.363   Mean squared error=0.17069
0.9 Quantile of absolute error=0.627
```



Checking the Model (first 2 plots)

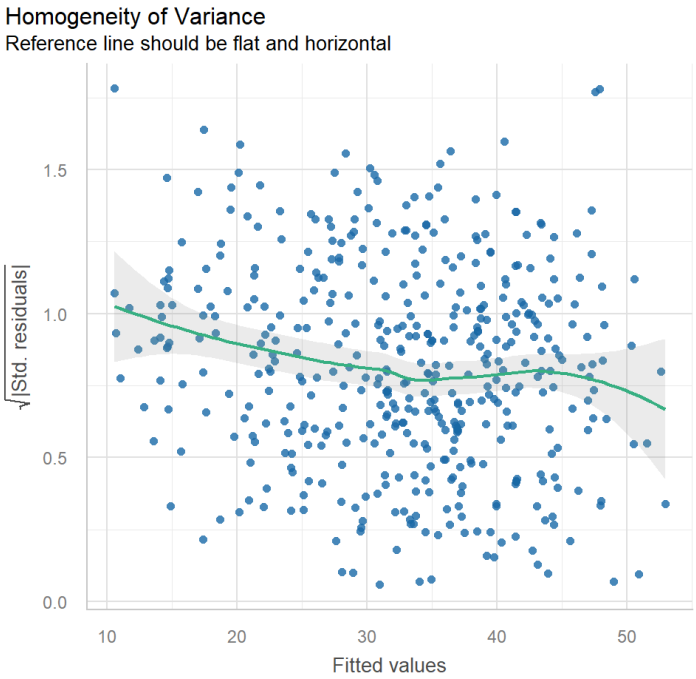
```
1 check_model(fit2_lm, check = c("pp_check", "linearity"))
```

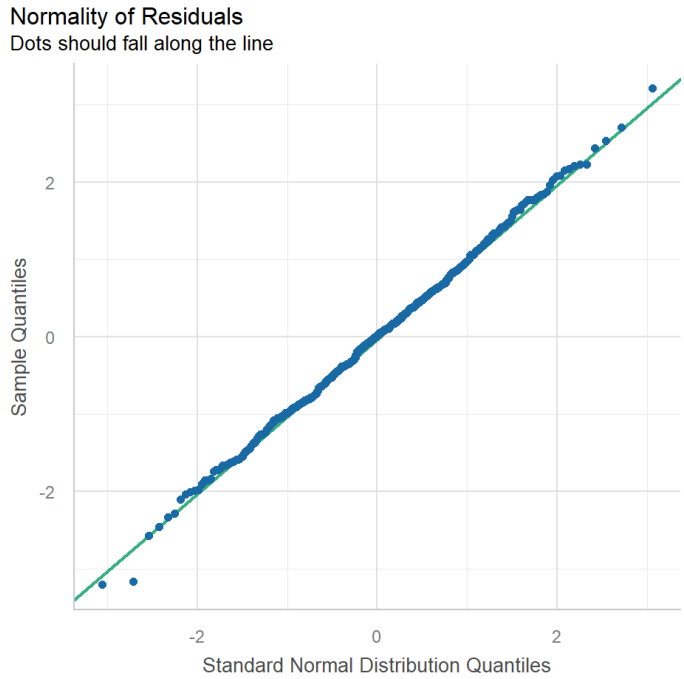




Checking the Model (plots 3-4)

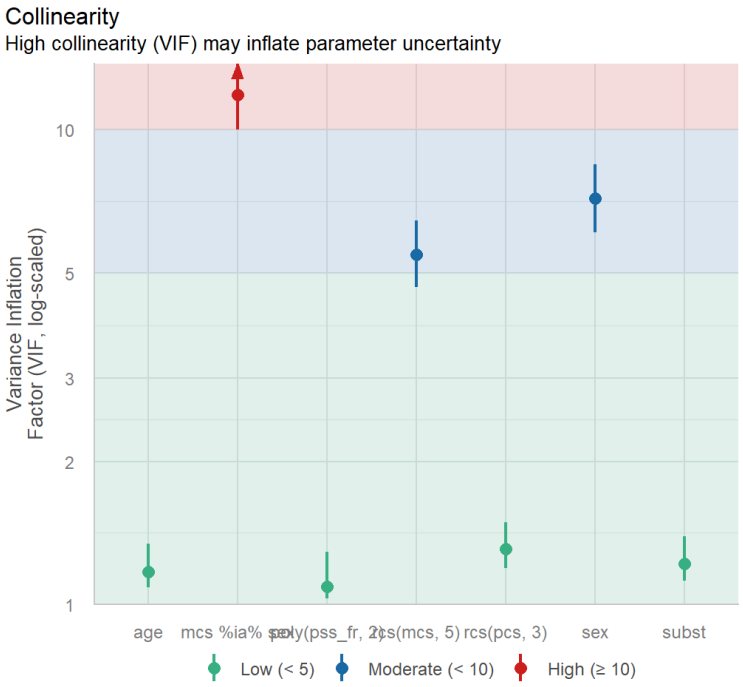
```
1 check_model(fit2_lm, detrend = FALSE, check = c("homogeneity", "qq"))
```





Checking the model (plot 5)

```
1 check_model(fit2_lm, check = c("vif"))
```



Checking Collinearity

```
1 check_collinearity(fit2_lm)
```

```
# Check for Multicollinearity

Low Correlation

           Term  VIF     VIF 95% CI adj. VIF Tolerance Tolerance 95% CI
    rcs(pcs, 3) 1.31 [ 1.19,  1.49]     1.14      0.76     [0.67, 0.84]
 poly(pss_fr, 2) 1.09 [ 1.03,  1.29]     1.04      0.92     [0.78, 0.97]
            age 1.17 [ 1.09,  1.34]     1.08      0.85     [0.74, 0.92]
          subst 1.22 [ 1.12,  1.39]     1.05      0.82     [0.72, 0.89]

Moderate Correlation

       Term  VIF     VIF 95% CI adj. VIF Tolerance Tolerance 95% CI
 rcs(mcs, 5) 5.46 [ 4.66,  6.43]     2.34      0.18     [0.16, 0.21]
```

Variance Inflation Factors

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

```
1 rms::vif(fit2)
```

```
          mcs           mcs'          mcs''         mcs'''            pcs 
    53.711079    4838.370091   12475.902431    2489.147506       5.521090 
          pcs'       sex=male mcs * sex=male         pss_fr       pss_fr^2 
     5.365910       7.163012      11.848760      15.657046      15.885078 
          age  subst=cocaine   subst=heroin                              
     1.172137       1.349517       1.383641                              
```

```
1 car::vif(fit2_lm)
```

```
                   GVIF Df GVIF^(1/(2*Df))
rcs(mcs, 5)    5.461428  4        1.236412
rcs(pcs, 3)    1.308116  2        1.069453
sex            7.163012  1        2.676380
mcs %ia% sex  11.848760  1        3.442203
poly(pss_fr, 2)  1.091682  2        1.022172
age            1.172137  1        1.082653
subst          1.217443  2        1.050418
```


Tests instead of plots?

- Never, ever, but …

```
1 check_heteroscedasticity(fit2_lm)
```

```
Warning: Heteroscedasticity (non-constant error variance) detected (p = 0.048).
```

```
1 check_normality(fit2_lm)
```

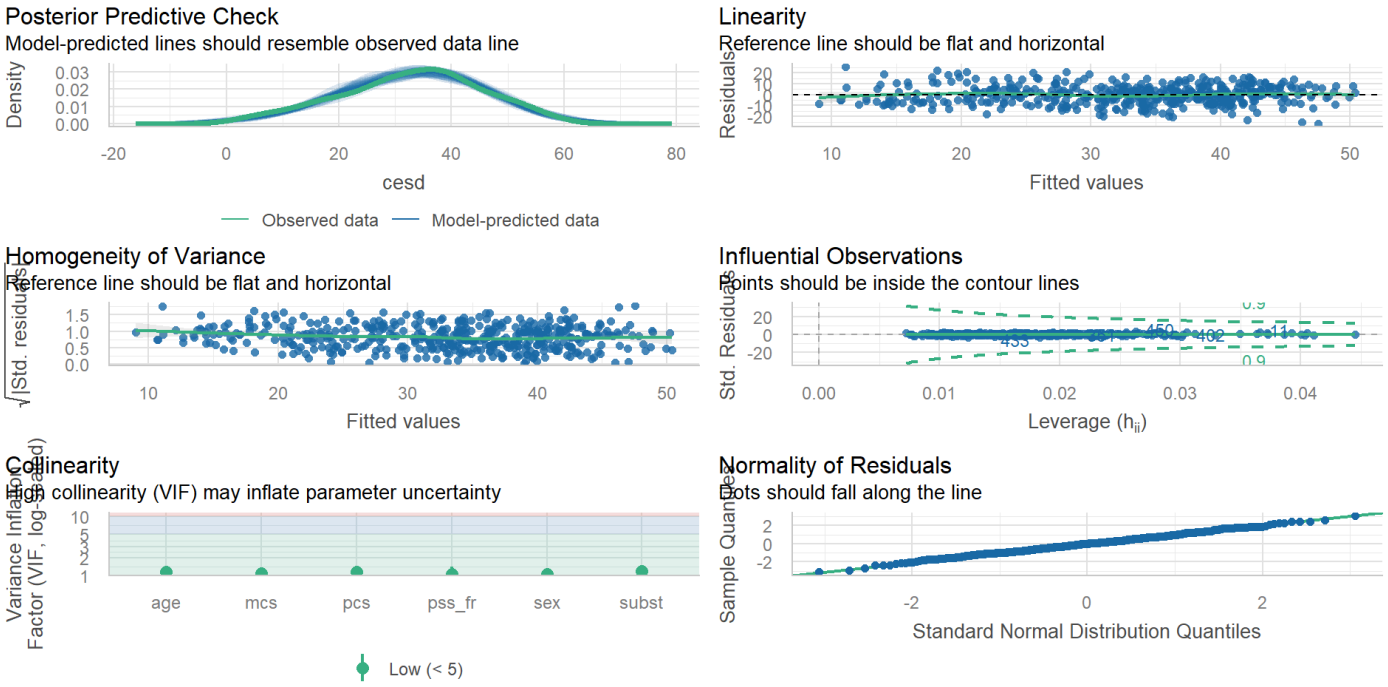
```
OK: residuals appear as normally distributed (p = 0.986).
```

```
1 check_outliers(fit2_lm)
```

```
OK: No outliers detected.
- Based on the following method and threshold: cook (0.9).
- For variable: (Whole model)
```

Checking model `fit1`?

```
1 check_model(fit1, detrend = FALSE)
```



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()`.

What's Coming Up?

- 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…