Linear regression 3

Outline

- Some questions from earlier
- Multiple independent variables and *(multi)collinearity*
- (For home consumption): the *likelihood-ratio* test.

Questions

The *z*-statistic in the Kendall τ_b test

I looked this up…

The test statistic τ_b lacks an easily-characterized distribution (there is no `pkendall` or `qkendall`). The standard way to compute the *p*-value for τ_b then is to convert it to a z-score.

R reports both τ_b and the *z*-score, though you don't need to report *z* since R does it for you on the way to computing the *p*-value.

The meaning of `~ (1/)`

In mathematical notation, ~ ("tilde") is sometimes read as:

- "is simulated by",
- "is a function of", or
- "is distributed according to".

For instance,

$$X \sim \mathrm{Bin}(p, n)$$

can be read as "x is binomially distributed (with n draws and success probability p).

The meaning of `~ (2/)`

R expands this a little bit to use this to write (first-class) objects it calls *formulae*, which can be passed to certain statistical functions (e.g., `t.test` and `wilcox.test`) and linear model functions like `lm`.

On the left-hand side of the `~`, we place the dependent variable; on the right-hand side, we place the independent variables, separated with +:

```
y ~ x + z
```

R interprets this (assuming X and Z are continuous) as $Y = \beta_0 + \beta_1 X + \beta_2 Z + \varepsilon$.

R automatically handles the intercept, dummy coding, etc.

The meaning of `~ (3/)`

It's important to understand: `~` is part of an *expression* in the R language: it doesn't have a numerical value. R interprets it depending on the context in which it is used.

Later on we'll see additional formulae syntax for

- disabling the intercept term,
- denoting *interactions* of independent variables,
- applying arithmetic operations directly to variables in the definition of the formula, and
- denoting *random effects*.

t-test example *sans* tilde

```
> x <- with(iris, Sepal.Width[Species == "versicolor"])
> y <- with(iris, Sepal.Width[Species == "virginica"])
> t.test(x, y)

    Welch Two Sample t-test

data:  x and y
t = -3.2058, df = 97.927, p-value = 0.001819
...
```

t-test example à tilde

```
> iris2 <- droplevels(
+     subset(iris, Species %in% c("versicolor", "virginica")))
> t.test(Sepal.Width ~ Species, data = iris2)

        Welch Two Sample t-test

data:  Sepal.Width by Species
t = -3.2058, df = 97.927, p-value = 0.001819
...
```

Tilde gotchas

While we often refer to the samples in a two-sample *t*-test or Wilcoxon test as *x* and *y*, they are both samples of *the dependent variable*; the independent variable is group membership. So there is no obvious connection between *x* and *y* in expressions like:

```
lm(y ~ x)
```

and

```
t.test(x, y)
```

Multiple regression

Linear regression

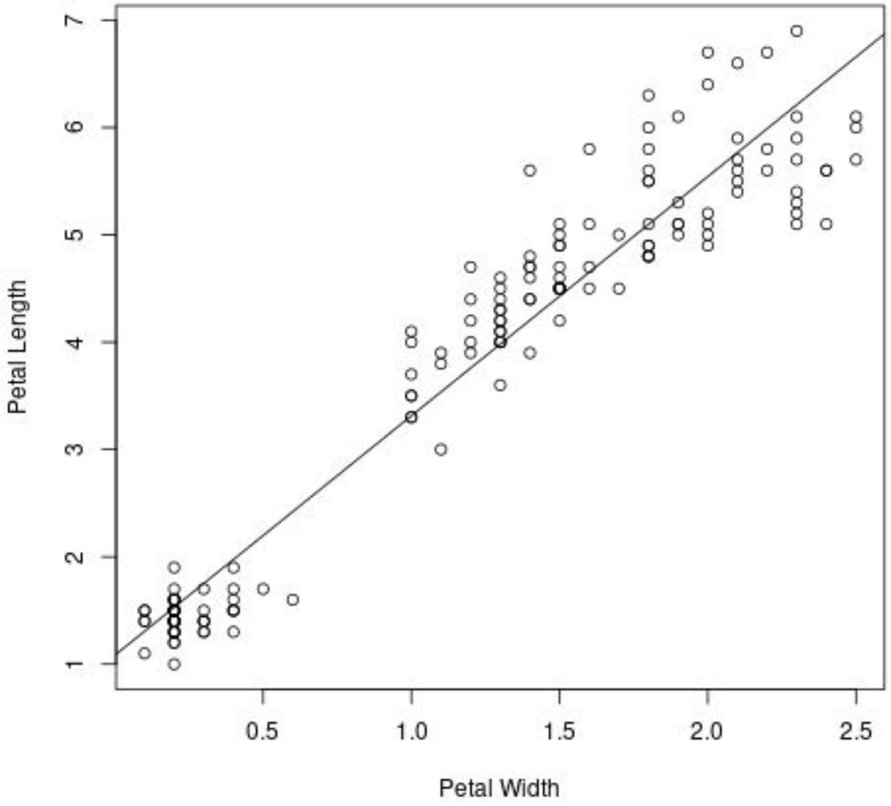
As mentioned, linear regression can be performed with multiple independent variables. This assumes

- *continuity*: independent variables must be expressable as continuous values,
- *variance*: independent variables must have non-zero variance,
- *linearity*: the dependent variable has a linear relationship between *each* independent variable (or is non-significant),
- *multivariate normality*: errors are normally distributed (or CLT),
- *homoscedasticity*: equal variance and standard deviation across IVs, and
- *no multicollinearity* (more on that in a second),

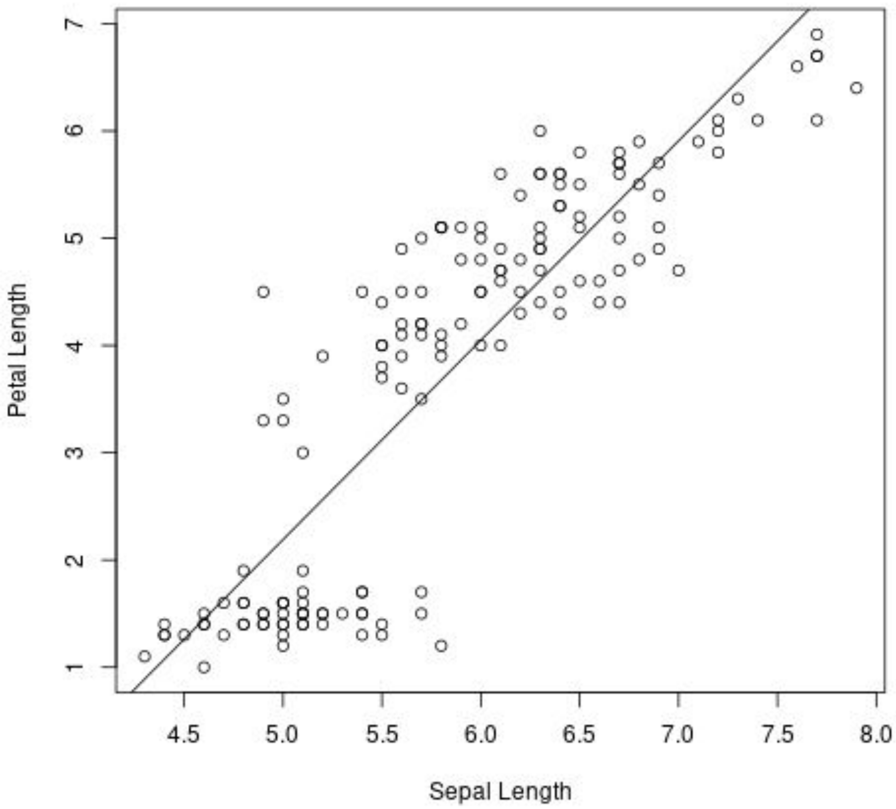
Example (1/)

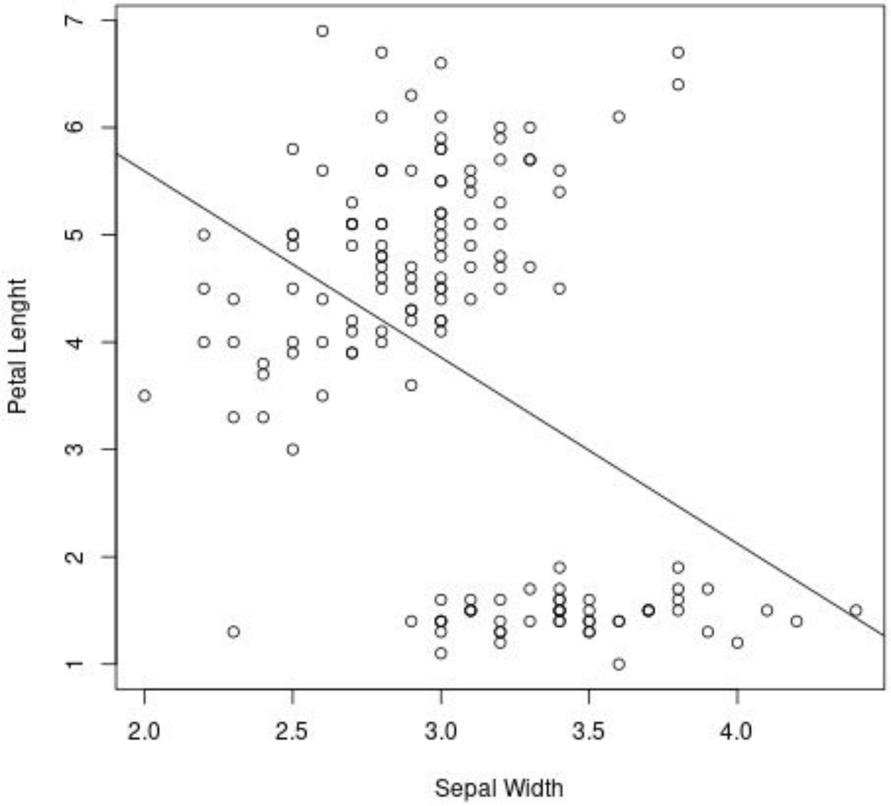
We'll try to predict one of the iris measurements (across all three species) using the other three.

Arbitrarily I chose `Petal.Length` as the DV and the remaining three measures as the IVs.



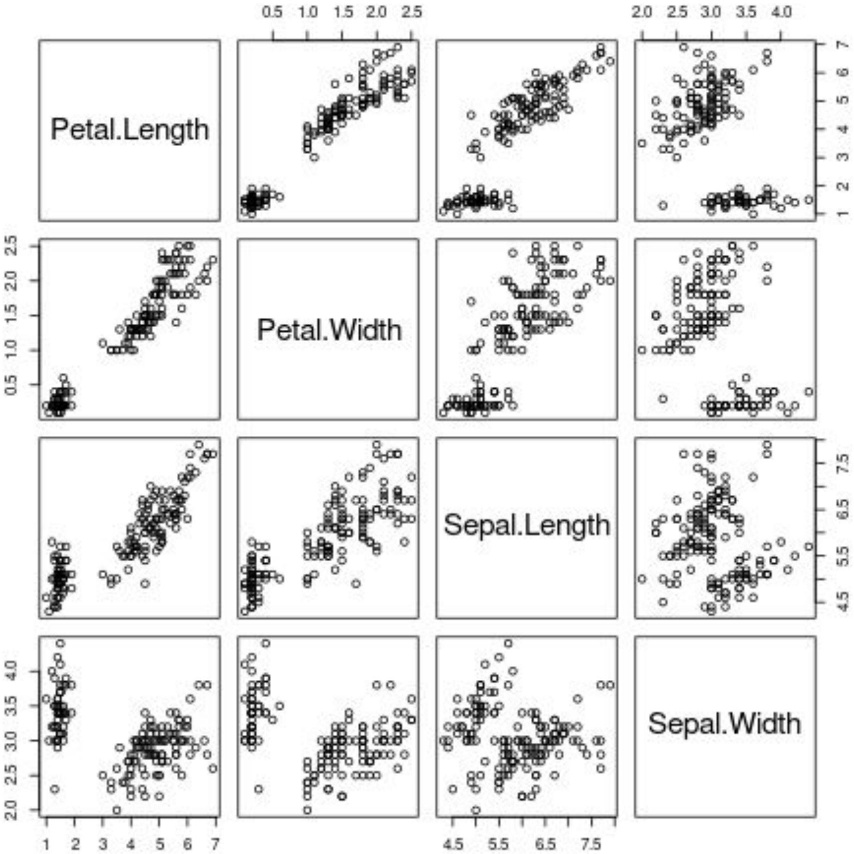
Petal Length
Petal Width


Petal Length
Sepal Length
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



Petal Lenght
Sepal Width
1
2
3
4
5
6
7
2.0
2.5
3.0
3.5
4.0

```
> pairs( ~ Petal.Length + Petal.Width + Sepal.Length +
Sepal.Width, data = iris)
```



Petal.Length
Petal.Width
Sepal.Length
Sepal.Width

```
> r <- lm(Petal.Length ~ Petal.Width +
+          Sepal.Width + Sepal.Length, data = iris)
> summary(r)

...
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.26271    0.29741  -0.883    0.379
Petal.Width   1.44679    0.06761  21.399   <2e-16
Sepal.Length  0.72914    0.05832  12.502   <2e-16
Sepal.Width  -0.64601    0.06850  -9.431   <2e-16
```

```
> r <- lm(Petal.Length ~ I(scale(Petal.Width)) +
+                          I(scale(Sepal.Width)) +
+                          I(scale(Sepal.Length)), data = iris)
> summary(r)

...
Coefficients:
                       Estimate Std. Error t value  Pr(>|t)
(Intercept)             3.75800    0.02604 144.302   <2e-16
I(scale(Petal.Width))   1.10280    0.05154  21.399   <2e-16
I(scale(Sepal.Length))  0.60377    0.04829  12.502   <2e-16
I(scale(Sepal.Width))  -0.28158    0.02986  -9.431   <2e-16
```

Variable entry

Three major styles:

- *Hierarchical*: experimenter uses hypotheses about the universe to build the regression and adds variables to model in order of importance
- *Forced entry*: all variables are added simultaneously
- *Stepwise*: experimenter adds (or removes) variables based on their *semi-partial correlation* with the outcome variable

Forced entry

All variables are added simultaneously to the model.

But if a very large number of variables are added at once, *overfitting*, in which extraneous independent variables are used to model error/noise, can occur.

One rule of thumb for categorical variables: "at least 15-20 cases per level".

Stepwise entry

In *step-up* entry, variables are entered one at a time, with some model criterion used to determine whether they remain or not. Alternatively, one can perform *step-down* entry, in which we begin with forced entry and remove variables according to a (negative) model criterion.

These are both implemented by the R function `step`.

This relies on experimenter-chosen criteria, which

- may not be appropriate,
- may not be interpretable, and which
- may not be sensitive to very small differences in *semi-partial correlation.*

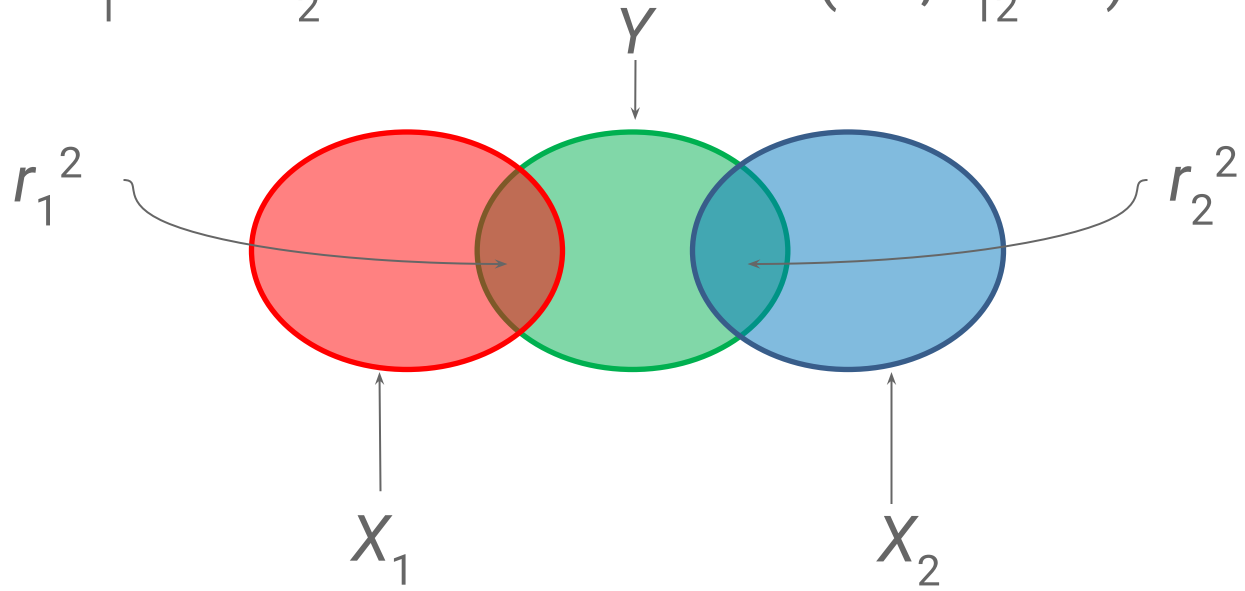
Accounting for variance

Recall that r^2 can be understood as the percentage of the variance shared by X and Y.

What if there are two predictors X_1 and X_2?

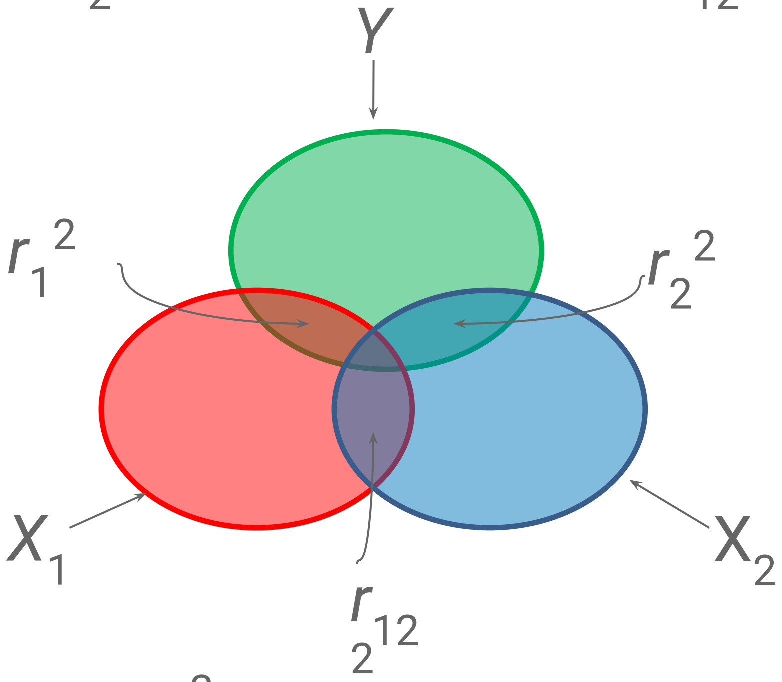
- If X_1 and X_2 are uncorrelated, then it is the sum their individual r^2s: $r^2 = r_1^{\ 2} + r_2^{\ 2}$.
- But otherwise, these terms must be adjusted.

X_1 and X_2 are uncorrelated (i.e., $r_{12} = 0$)



$$r^2 = r_1^2 + r_2^2$$

X_1 and X_2 are correlated (i.e., $r_{12} \neq 0$)



$$r^2 = B_1 r_1 + B_2 r_2$$

where B is an adjustment.

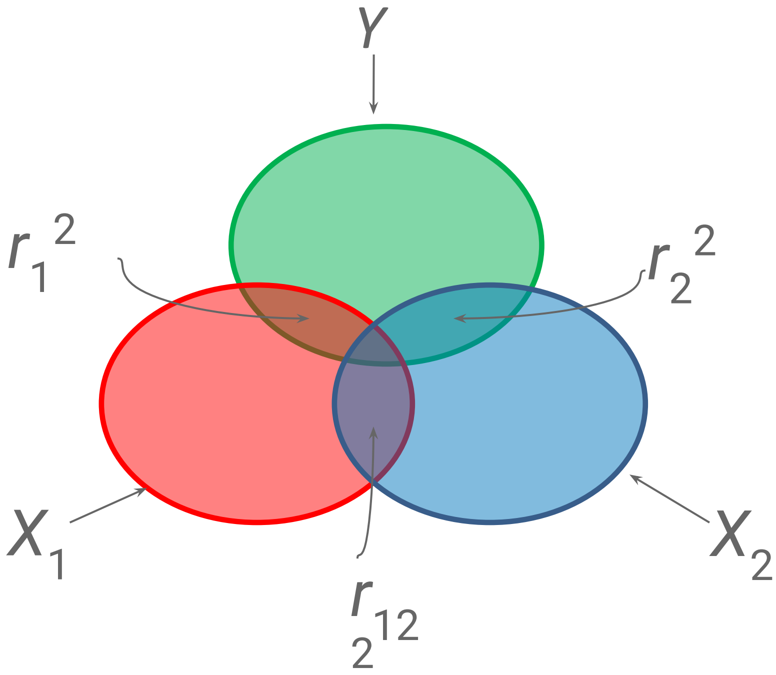
Partial and semi-partial correlation

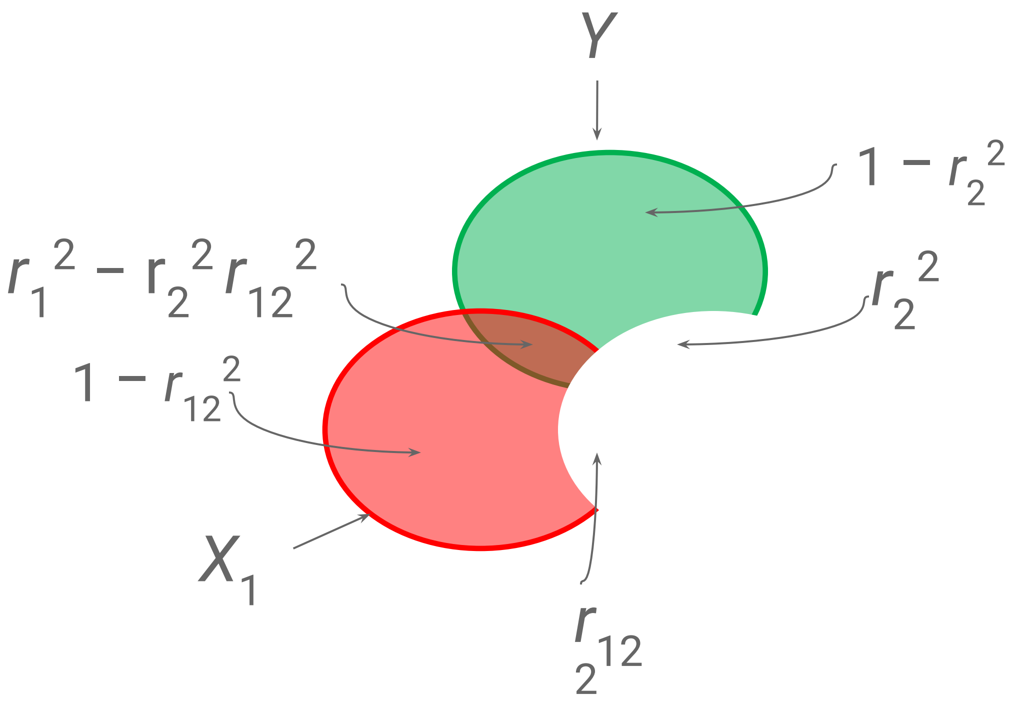
Partial correlation measures the relationship between two variables, controlling for the effect that a third variable has on them both.

Semi-partial correlation measures the relationship between two variables controlling for the effect that a third variable has on only one of the others.

In the context of regression, semi-partial correlation measures

- the relationship between a predictor and the outcome, controlling for the relationship between that predictor and any others already in the model, or
- the unique contribution of a predictor to explaining the variance of the outcome.


Y
r_1^2
r_2^2
X_1
X_2
r_{12}
2



$$r_{1.2}^{2} = (r_1 + r_2 r_{12}) \,/\, \sqrt{[(1 - r_2^{2})(1 - r_{12}^{2})]}$$

Residualization

"Controlling for X_2" is the same as:

- "removing the effect of X_2",
- "holding X_2 constant" or
- *residualization*, using X_2 to predict X_1 and extracting the *residual*, the portion of X_1 uncorrelated with X_2.

(Multi)collinearity

Multicollinearity exists when two or more predictors are highly correlated as measured by Pearson's *r*. Multicollinearity

- violates the statistical assumptions we use for hypothesis testing, and
- when extreme, can cause the parameter estimation technique to fail.

NB: The *multi*- bit in *multicollinearity* doesn't really denote anything in particular, other than "non-trivial collinearity between two or more independent variables."

Other ways of measuring multicollinearity

- *Variable inflation factor* (VIF; Davis et al. 1986; `vif` in the `car` package): "the inflation in size of the confidence ellipse or ellipsoid for the coefficients of the term in comparison with what would be obtained for orthogonal data"; e.g., 1.6 would mean that the confidence intervals are 1.6x larger than they would be if the data had no multicollinearity.
- κ ("kappa"; Belsey et al. 1980): κ > 10 is considered to indicate non-trivial multicollinearity.

Residualization (1/)

Residualization addresses multicollinearity by creating new *orthogonal* independent variables. We create said variables by fitting simple linear models and extracting the residuals. E.g., if *a* and *b* are two independent variables we might do:

```
> b.res <- residuals(lm(b ~ a))
> stopifnot(cor(a, b.res) == 0)
> r <- lm(y ~ a + b.res)
```

Note that one independent variable acts as a baseline.

Residualization (2/)

When there are two or more multicollinear variables, this has to be performed iteratively. Suppose we have three collinear independent variables *a*, *b* and *c*:

```
> b.res <- residuals(lm(b ~ a))   # As before.
> stopifnot(cor(a, b.res) == 0)   # As before.
> c.res <- residuals(lm(c ~ a + b.res))
> stopifnot(cor(a, c.res) == 0)
> stopifnot(cor(b.res, c.res) == 0)
> r <- lm(y ~ a + b.res + c.res)
```

Questions? Please take them to email, or Slack.