Working with models

Model building

Including predictors

Choosing a response distribution

Setting priors

The first chapter of this book linked Bayesian statistics to rational decision making. You recall there was much a do about incorporating prior knowledge. In Bayes' formula, prior knowledge is a central component and was expressed as a probability. If this is so awsome as it sounds, why, you may ask, wasn't there any mentioning of priors in the following chapters. I made a difficult choice, here.

All modelling commands used so far, understand specification of prior knowledge. At the same time, these commands provide useful default priors. The defaults are *non-informative priors*. They are formally specified, but have negligible impact on the posterior distribution. Consider the case BrowsingAB. In [CLM], we estimated the linear slope between age and ToT.

```
attach(BrowsingAB)
```

The default prior specification can be found in the manuals of rstanarm (or brms). Or, you use the following command on the model object `M_2`:

```
detach(BrowsingAB)
```

- extract default prior from M_2
- extreme example: guessing a persons IQ, without a measure (100+-15)

People had been using the data to come to conclusions, but they also relied on their own convictions. This sounds subjective and idiosyncratic, even. Is that still research, strictly? In fact, some have labelled Bayesian statistics subjective, and therefore inadmissible, for the concept of priors. What has been overlooked by those is:

1. Frequentist statistics has priors, too. They are just always flat (non-informative).
2. Non-informative priors in Bayesian estimation are routinely used, too.
3. By putting prior knowledge into a formula, it is objective and becomes criticizable. Every reviewer, stakeholder or follow-up researcher can inspect the chosen prior and follow it, or not. When research is published in a reproducable way, people can even reproduce the estimation using their own priors, when disagreeing.

Model convergence

Throughout this book I have fooled you into thinking that the regression engine is something where you put your data and model in and you get some tidy MCMC chains as output, like putting gras into a coow produces milk. MCMC chains are random walks and they are not at all as cows. MCMC chains are like a bulk of young kitten, everyone on their own route, bumping into things or getting stuck in corners. Stan is very good with kitten and safely guides their path through the unknowns of a multidimensional parameter space. Most of the time.

Pathologies of MCMC chains:

1. too few cycles
2. chains plateau for longer periods
3. chains do not arrive at similar distributions
4. chains wipe over tiny gaps in the distribution

- MCMC convergence
- eff sample size
- Tuning the Stan engine

Model criticism

- residual analysis
- linearity
- colinearity
- posterior predictive checks

Gelman, a, Gelman, a, Meng, X.-L., Meng, X.-L., Stern, H., & Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. Vol.6, No.4. Statistica Sinica, 6(4), 733–807. http://doi.org/10.1.1.142.9951

Model evaluation

The tenor of this book on new statistics is quantification and up to this point all interpretation focussed on the model coefficients: under uncertainty, the center estimate serves as a best guess for effect size and by the credibility limits we express how good our best guess is. Researchers trained in classic statistics often completely ignore coefficient tables. Their perspective merely is whether an effect stands out from the noise (i.e., uncertainty) enough to be called statistically significant. This practice has been criticized for decades and good reasons. One fundamental criticism being that the concept of statistical significance artificially divides the world in significant and non-significant results. That stands in stark contrast to the intuitions that reality is continuous and research is incremental.

What many classic researchers seem to be unaware of, in addition, is that the approach of statistical significance, typically employing p-values, is a special case of a much broader class of statistical procedures called model selection. More specifically, an analysis of variance for the purpose of rendering the difference between two groups of observations is a basic comparison of two models: the alternative model is associated with the hypothesis the researcher has in mind and carries a coefficient to represent the effect. The null model assumes that there is no effect and is the alternative model reduced by the respective coefficient.

$$H_1 : y_i = \beta_0 + \beta_1 x_{1i} H_0 : y_i = \beta_0$$

Another way to put it is that H_0 is a special case of H_1, where the slope coefficient is fixed to zero, $\beta_1 = 0$. If one model is a special case of another, these models are called *nested models*.

All classic tests only work with nested models. The famous F-test estimates both models, calculates the residual sum of squares under both models and evaluates whether the more general model sufficiently reduces the residuals, relative to the nested model, to be called statistically significant. As we will see in the remainder of this chapter, comparing models relative to each other has useful applications beyond classic hypothesis testing, in particular:

1. Models that contain superfluous predictors have lower predictive power compared to a more parsimonious model.
2. When the main purpose is predicting future observations with optimal accuracy, *model pruning* is a procedure for selecting a subset of parameters with the optimal forecasting accuracy.

3. In chapter [REF] we used graphical methods to choose an appropriate error distribution for reaction time measures. By comparison models, we can identify the optimal error distribution for the data at hand.

As it will turn out, the forecasting accuracy of a model depends on two counteracting aspects: *model fit*, that is how neatly the model aligns with the data and *model parsimony.* Generally spoken, adding a parameter to a model gives you an advantage in fit, but potentially reduces parsimony, resulting in less-than-optimal forecasts. Even classic ANOVA contains both elements: the F-statistic represents model fit by the reduction in residuals and parsimony enters the F-test as degrees of freedom, i.e. the number of free parameters.

This chapter will not re-iterate on classic tests, but will approach model comparison from the perspective of forecasting accuracy. One very practical reason for that is, that classic tests *only* apply to nested models, whereas there are many situations, where comparing non-nested models is very useful.

The remainder of this chapter will first elaborate on model parsimony and demonstrate that parsimony is crucial for forecasting accuracy. Then, a generic procedure for measuring forecasting accuracy of models will be outlined, the *leave-one-out cross validation.* Then we introduce a very practical class of statistics for model comparison called *information criteria.* As we will see, information criteria apply to a much wider class of models, including non-nested models, MLM and GLM, but also non-linear models or psychometric models.

Forecasting accuracy

At several occasions in this book, we used the `predict(model)` function to produce a plot for model criticism, where fitted lines are shown next to the original data. First, a note on terminology is in order: In common language, a prediction is an informed idea about future events, hence, observations we have not yet made. This is strictly not what the `predict` function does, as it produces the best guess for the observations we already have in the data set. This best guess is what the model "thinks" is the true underlying property, minus the error. For that reason, other authors have suggested to call predicted values *retrodictions.*

Goodness of fit

- RMSE
- deviance
- total deviance

The principle of parsimony

Occam's razor sounds bloody business. In words of Bertrand Russell it says:

> if one can explain a phenomenon without assuming this or that hypothetical entity, there is no ground for assuming it.

Why not? And mind how the speaker carefully avoids to condempt unnecessary hypotheses right away. Occam's razor is sometimes taken as what mathematicians call an axiom. Axioms are assumptions that don't have another Why. Often, these axioms are intuitive, like: Every natural number has a successor."

Occam's razor, or the *parsimony principle*, is intuitive, because people reportedly assume things for their own grounds a lot. It seems fully in order to request a counter balance. Or staying with the original metaphor, a blade is needed that prunes exuberant creativity of the researcher.

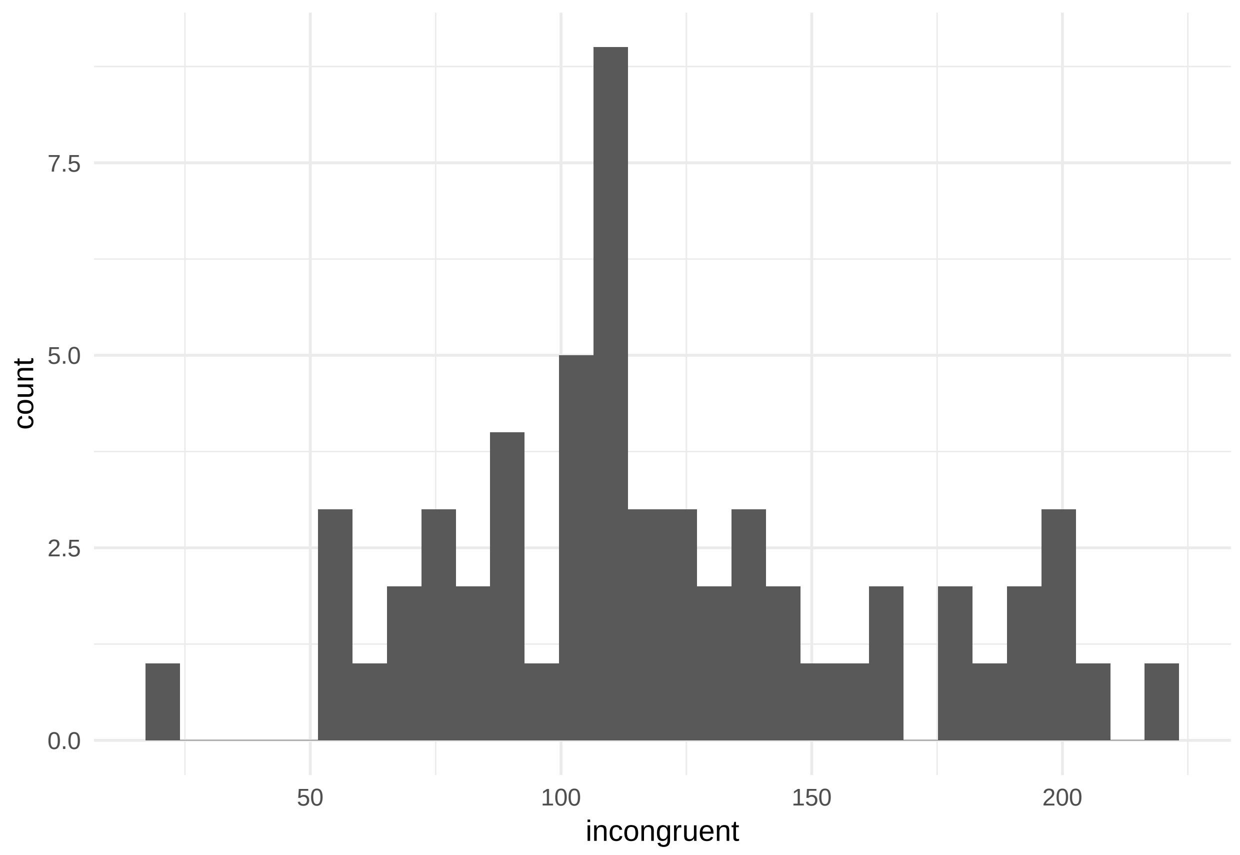
The Popperian Philosophy of Science elevates the principle of parsimony to a driving force in scientific progress even. This goes about like this: Popper's fundamental criticism was that researchers aimed at proving their theories by gathering a lot of confirmatorty data. Popper argued, that you can never prove a universal scientific law, because that would require you to gather all possible outcomes of that law. Popperians requests that a theory must be *falsifiable to be called a scientific theory.* As it turns out, a required condition for a theory to be falsifiable, is that it makes *testable predictions.* In turn, a testable prediction is one where you can find *counter examples.* This is best explained by example:

The infamous Stroop effect can be observed when people see color words, but have to respond the ink color of the typeface. Then, as has been reported abundantly, people respond with a delay, when the word and the ink color are incongruent, i.e. they do not match. Multiple theories emerged from decades of research on this, not strange, but extremely robust effect. Here are some:

Incongruent sources of verbal information compete for processing capacity of a certain module in the brain, therefore: Everyone who has a brain is falling for the Stroop effect. This is roughly what many cognitive theories on the Stroop effect assume in more sophisticated ways. At the same time, the Stroop effect has been shown to happen in hundreds of studies, and I do not know of any counter-examples. Is this theory a certain endeavour? Not so fast! The above theory speaks of everyone, not in every study. That means specifically, that the Stroop effect is apparent for every single one of the partiucipants, not just en grosse. Unfortunately, most studies on the Stroop effect only report the population-level group effects, but not participant-level effects. We can test this theory by taking a closer look at teh Stroop data set [REF]. Are all participant-level coefficients positive in the expected direction? Yes, they are. Although, there exist individuals who come close to be unaffected.

```
attach(StroopMW)
```

```
Scores <-
  D_stroopmw %>%
  distinct(Part) %>%
  bind_cols(
    ranef(M_1_exg) %>%
      filter(fixef == "Conditionincongruent") %>%
      select(ranef = center)
      ) %>%
  mutate( fixef = fixef(M_1_exg) %>%
            filter(fixef == "Conditionincongruent") %>%
            select(fixef = center) %>%
            pull(1)
          ) %>%
  mutate(incongruent = (fixef + ranef) * 1000)

Scores %>%
  ggplot(aes(x = incongruent)) +
  geom_histogram()
```



```
detach(StroopMW)
```

Another observer of the Stroop effect could draw more on the obvious lack of obedience that participants show. They were not asked to read a single word in this experiment, but apparently they do. This lack of obidience towards instructions is what cognitive experimentalists often call bottom-up processing. It is commonly interpreted as involuntary cognition triggered by sensations. In brief this theory goes: *Reading is highly over-learned and whenever literate persons see a written word, they read it.* This theory proposes that we are so attached to words that we cannot read not. Does always it? Here is my one counter-examples. I remember the day, when at my workplace everyone got a major upgrade on their computers. One highly praised feature of the new user interface was a ubiquitous search function, which was placed in the programs menu, just left of the power-down button. The other day, a colleague entered my office and told me that he had just discovered a usability problem with the search function. Following a habit, he had entered a search term and pressed the button right next to it. This button clearly said “Shut down computer”. He had not read. This theory is falsified.

Falsifiability is a continuum and depends on universality and precision of a theory. For *universality*, we have just seen an example. The theory of overlearned reading is a rather universal and anecdotal data made it fail. Popper concluded that a theory is more falsifiable when it is more universal. In order to rescue the theory, we have to make it more specific, such as: *People always read when forced to see a word.* Are they? Take five minutes to stare at the following letters. Notice the moment when you stop reading it. Notice the many moments, your thoughts go astray, but always return to the letters. Notice the moment when you stop seeing a word, at all.

```
W O R D
```

In the previous chapter Falsifiability also depends on *precision* of a theory; here is an example: Many theories on the Stroop task are nominating a certain information processing pathway as a cause and predict *a* delay in the incongruent condition. Any of those theories is consistent with the above results. A funny fact is that most cognition researchers would call the Stroop task a complex task. That somehow implies longer processing times and, perhaps, one should only call it Stroop effect when it sits north of 50ms. With precision comes falsifiability: one participant can make such a theory fail.

Comparing two theories by their parsimony is not easy, as it requires a good deal of formalization. Statistical models are formal and there seems to exist a straight-forward definition for *model parsimony: of two models, the one with fewer independent parameters is more parsimonous and the one with more is more opulent.* In single-level linear models, the nominal number of parameters is precisely the number of coefficients in the likelihood plus another parameter for the error distribution, such as σ^2.

Model opulence

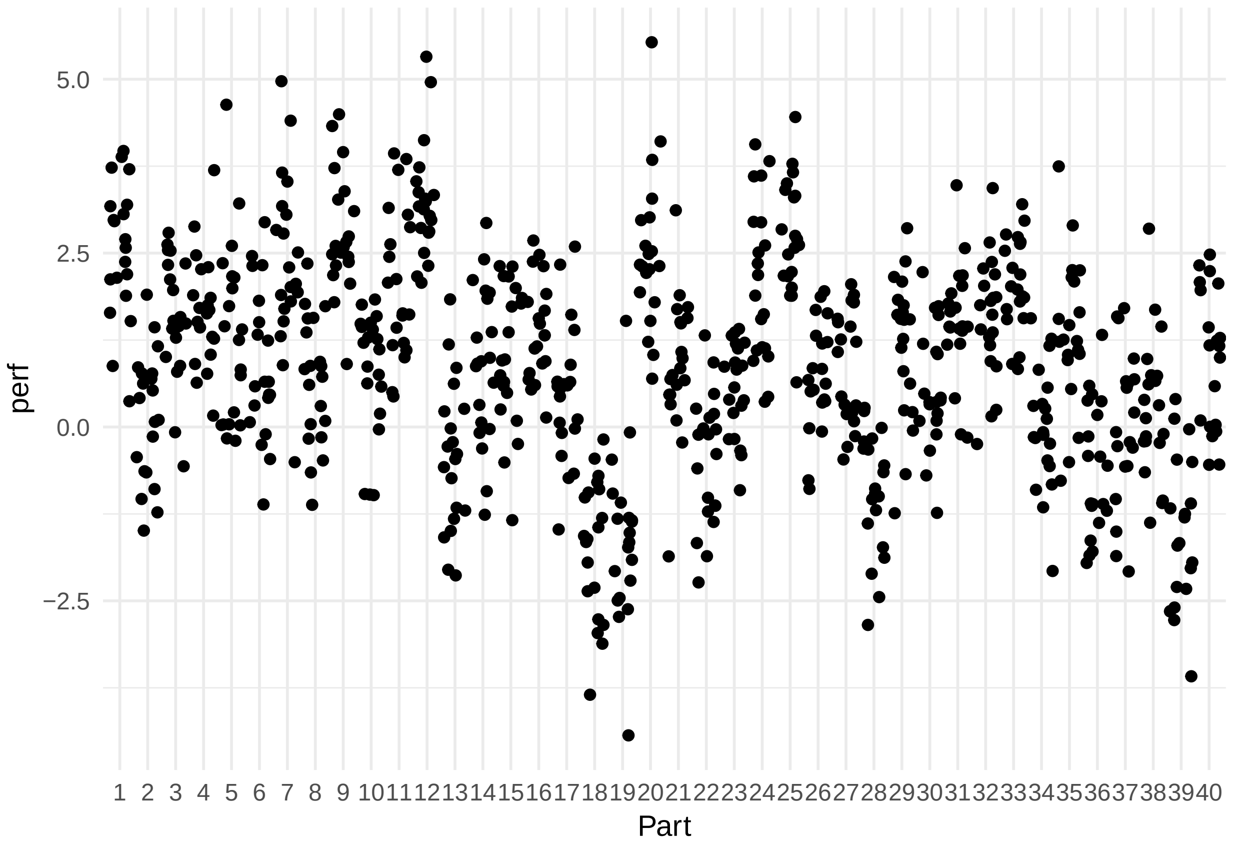
In single-level linear models all parameters are independent and counting the nominal parameters is a valid estimate for model opulence. In multi-level models there can be cross speak between levels and parameters influence each other. It is no longer adequate to count all parameters for model opulence.

As explained in [REF: Shrinkage], participant-level parameters in multi-level models are not independent. By estimating participant variance alongside, the individual random coefficients are influencing each other. Not totally unlike human cognition bottom-up and top-down influence takes place: the population level learns from the participant level how much clumping there is and participants with extreme outcomes (or little data) are adjusted towards the center of the population-level distribution (shrinkage). Let us see that on a simple example. The following simulation function creates two-level data sets with repeated measures of participant performance. We will see down the line, how module opulence varies with the model, but also with properties of the data.

```
sim_1 <-
  function(N_Obs = 20, N_Part = 40, beta_0 = 1, beta_0_sd = 1, error = 1, seed = 42){
    set.seed(seed)
    Part <- data_frame(Part = as.factor(1:N_Part),
                       beta_0p = rnorm(N_Part, 0, beta_0_sd))
    expand_grid(Part = Part$Part, Obs = as.factor(1: N_Obs)) %>%
      left_join(Part, by = "Part") %>%
      mutate(perf = rnorm(N_Obs * N_Part, beta_0 + beta_0p, error))
  }
```

```
attach(Chapter_WM)
```

```
sim_1() %>%
  ggplot(aes(x = Part, y = perf)) +
  geom_jitter()
```



Although we can easily spot the participant-level effect, as a baseline for model opulence we use a grand mean model with two completely independent parameters β_0 and σ. Because they are independent, the effective number of parameters is equal to the nominal parameters in the model.

```
M_1 <-
  sim_1() %>%
  stan_glm(perf ~ 1, data = .)

M_1
```

In multi-level models, parameters are not independent. The *effective number of parameters* is smaller than nominal and *needs to be estimated.* At this point suffice it to say that the *Widely Applicable Information Criterion (WAIC)* introduces such an estimate [REF] of effective parameter (opulence). Calling the function `waic()` on rstanarm models produces, among others, the statistic p_waic. Of course, it also works on models with independent parameters, such as as the grand mean model:

```
waic(M_1)$p_waic
```

```
## [1] 1.94
```

As expected, the estimated number of parameters is close to the nominal two. Note that p_{waic} is an estimate on the draws of the MCMC random walk and this is why the result will never be fully exact (but arbitrarily exact by running longer MCMC chains). Next, we introduce a participant-level effect to the model. We will do that as a fixed effect, first, allowing no top-down influence, the nominal number of parameters of this model is 41:

- intercept
- 39 differences
- residual standard deviation

```
M_2 <-
  sim_1() %>%
  stan_glm(perf ~ 1 + Part,
                data = .)
```

```
waic(M_2)$p_waic
```

```
## [1] 3.44
```

The opulence estimate is only slightly smaller than the nominal number of parameters. The next model introduces a genuine random effect, where participant-level coefficients are no longer unconstrained. The nominal number of parameters is 43:

- population intercept
- population standard deviation
- 40 participant intercepts
- residual standard deviation

```
M_3 <-
  sim_1() %>%
  stan_glmer(perf ~ 1 + (1|Part),
                data = .)
```

```
waic(M_3)$p_waic
```

```
## [1] 38.7
```

Replacing the fixed effect by a random effect introduced two more parameters, but model opulence actually goes down a bit. This is a subtle shrinkage effect at work. Next, we simulate an extreme case of an extremely tight population-level variance. Under such circumstances individual differences are negligible and the population mean rules. The estimated model becomes more parsimonious, almost approaching the parsimony of the grand mean model.

```
M_4 <-
  sim_1(beta_0_sd = 0.001) %>%
  stan_glmer(perf ~ 1 + (1|Part),
                data = .)
```

```
waic(M_4)$p_waic
```

```
## [1] 4.5
```

Finally, the strength of bottom-up influence depends on how certain those participant-level coefficients are . Up to this point we have simulated a solid 20 observations per participant and received determined participant intercepts. Reducing to five measures per individual results in considerably lower model opulence.

```
M_5 <-
  sim_1(N_Obs = 5, N_Part = 40) %>%
  stan_glmer(perf ~ 1 + (1|Part),
                data = .)
```

```
waic(M_5)$p_waic
```

```
## [1] 33.5
```

```
detach(Chapter_WM)
```

To sum it up: this section introduced the effective number of parameters as a measure for model opulence (the opposite of model parsimony). We have seen how opulence depends on the model, but also strongly on the structure of data. This means that a researcher cannot determine upfront what would be a model with reasonable parsimony. But why should we actually care about parsimonious models? The next section we demonstrates what sounds counter-intuitive, at first: a more opulent models usually fits the existing data better, but can be worse at forecasting future observations.

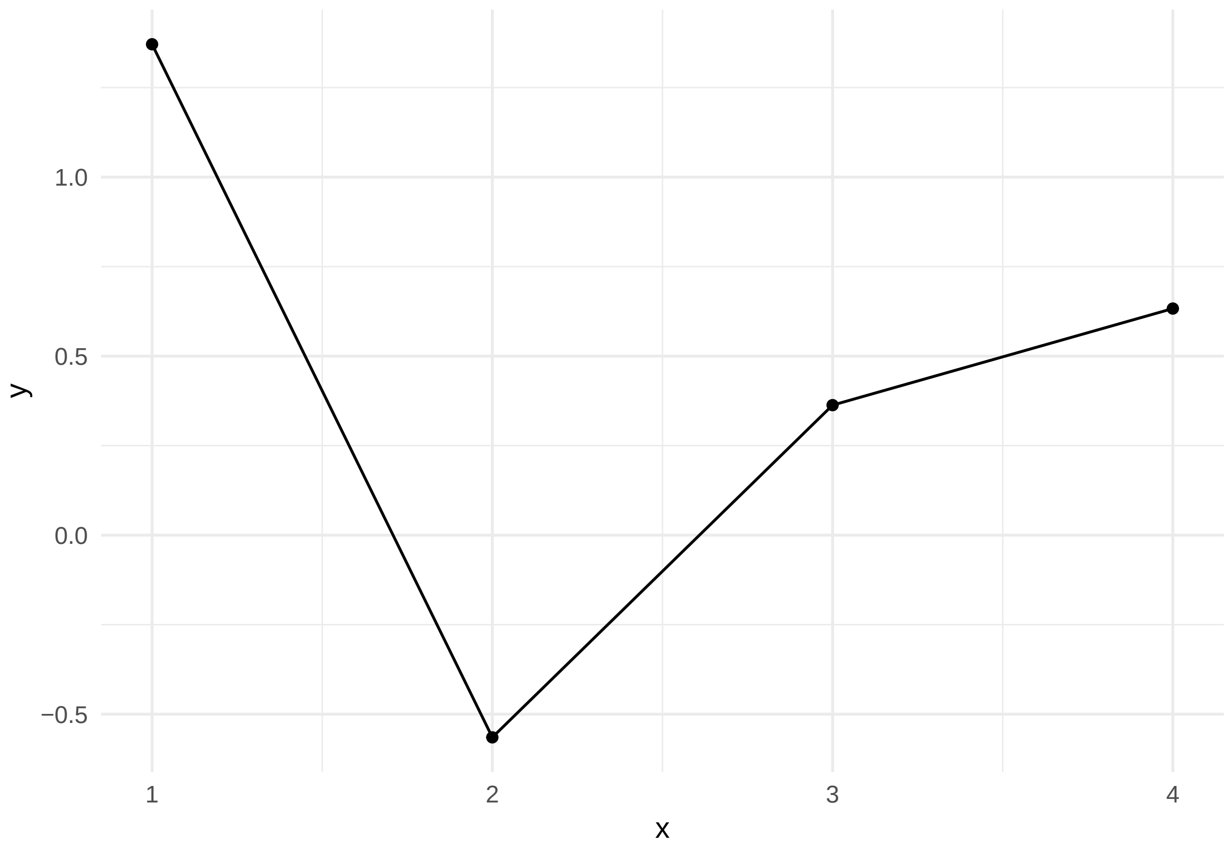
Too opulent: over-fitting

Model fit and forecasting accuracy should never be confused because they have quite the opposite relation with parsimony. Forecasting future observations is a completely different business, as we will now see. In the following, we will use a small simulated data set and impose two models: a very parsimonious model that matches the data generating process and a very opulent model, that carries more parameters than there are observations, even. As it will turn out, the parsomonious model produces much tighter retrodicted values, but performs considerably worse in a forecasting.

The following simulation draws two variables, x_i and y_i in such a way that they are completely unrelated. are totally unrelated to the predictor x_i (with levels 1 to 4):

```
attach(Chapter_WM)
```

```
sim_2 <- function(n) data_frame(x = 1:n, y = rnorm(n))

set.seed(42)
D_4 <- sim_2(4)

D_4 %>%
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  geom_line()
```

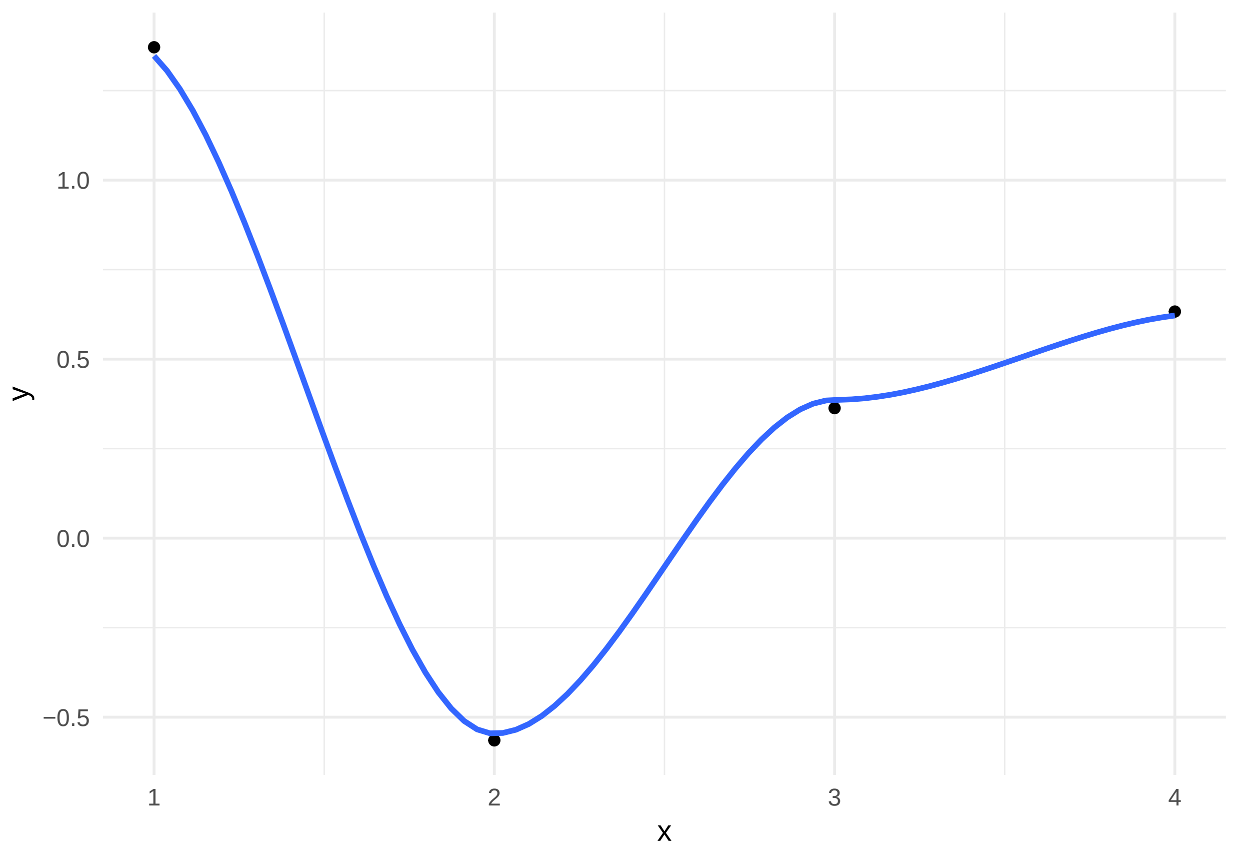


A naive observer (with some fancy theoretical expectations) may come to the idea of a curvi-linear relationship between x_i and y_i, similar to the one we have used to capture the Uncanny Valley effect in the emotional judgment of robot faces [REF]. The presence of a parabolic element and a shoulder suggests a polynomial model of third degree. This model has five nominal parameters:

- four polynomial coefficients
- residual standard error

```
M_6 <-
  D_4 %>%
  stan_glm(y ~ poly(x, 3), data = .)
```

```
D_4 %>%
  mutate(pred_M_6 = predict(M_6)$center) %>%
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(aes(y = pred_M_6), se = F)
```

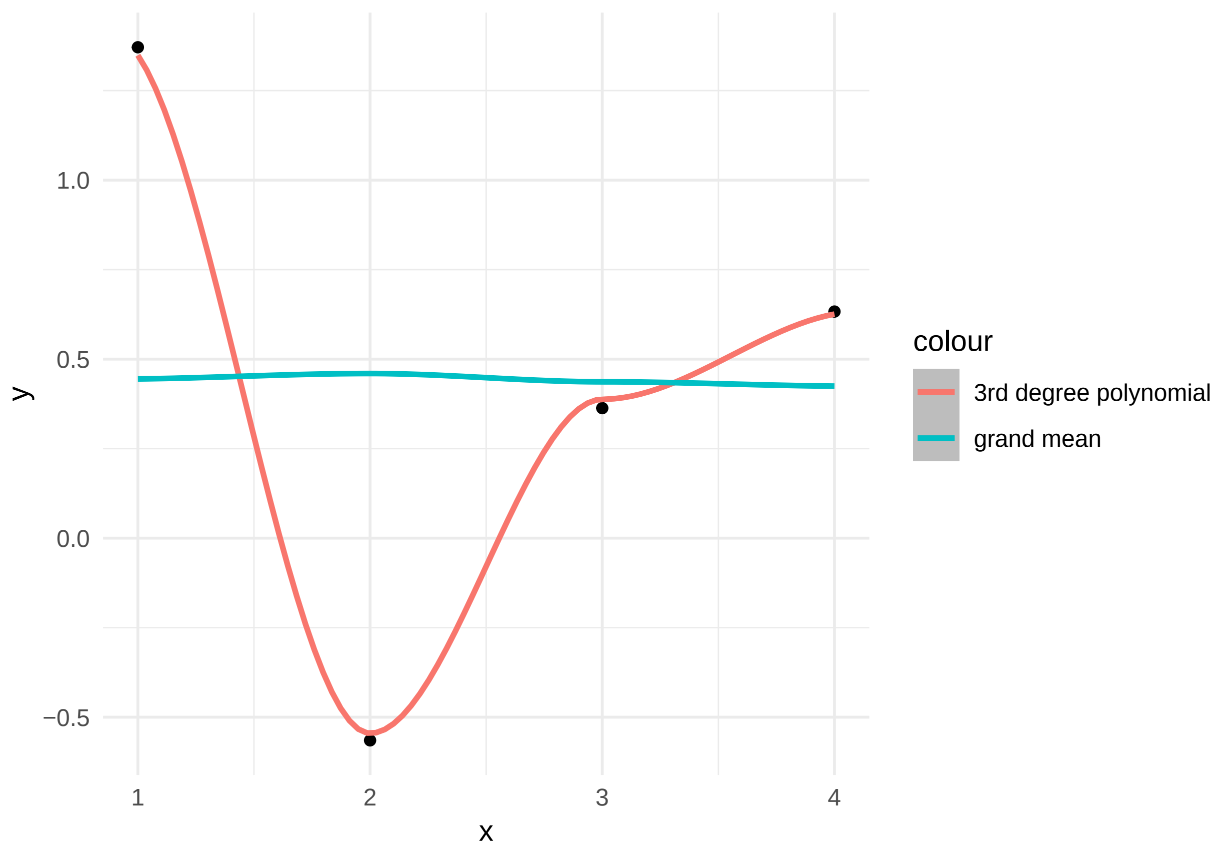


The fit of the model is almost perfect. But, remember that these are retrodictions, not forecasts. A more critical observer would suggest to start with a parsimonious grand mean model, as it happens to be the true data simulation process:

```
M_7 <-
  D_4 %>%
  stan_glm(y ~ 1, data = .)
```

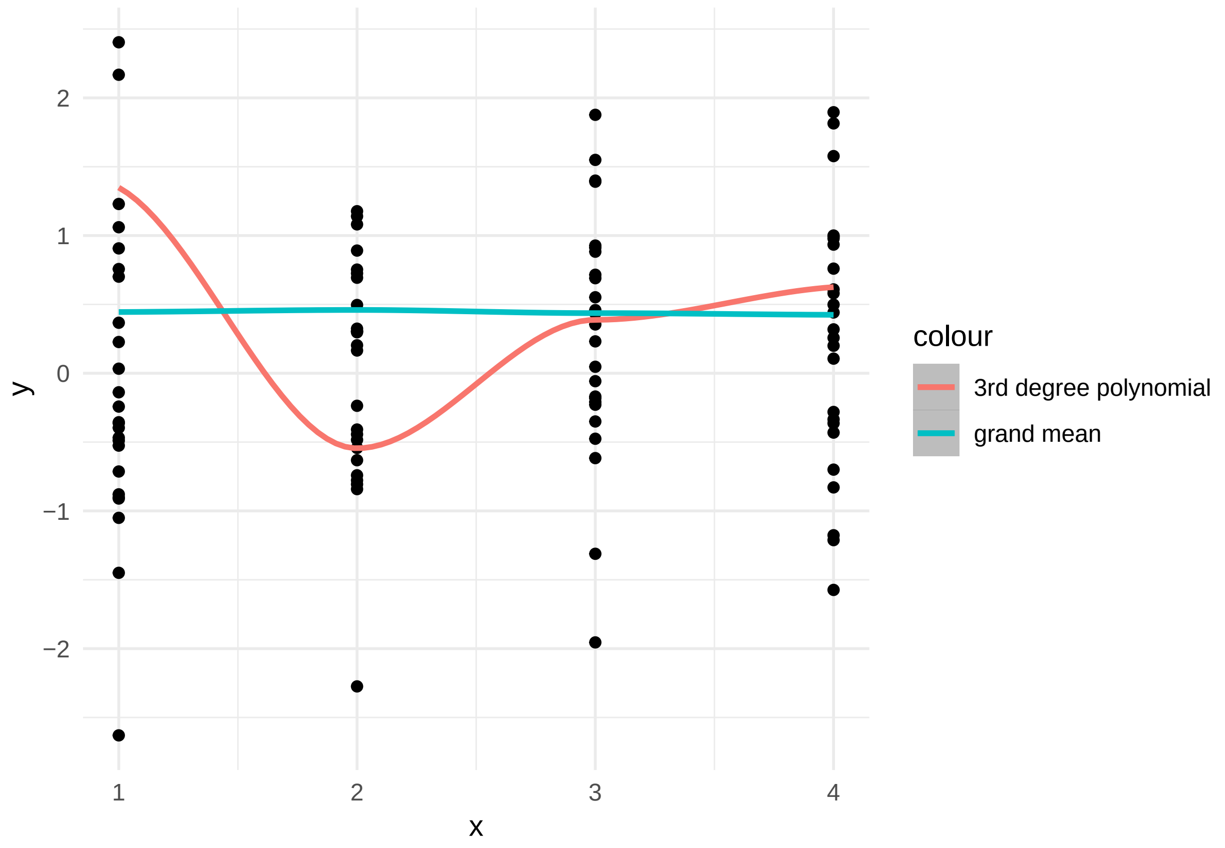
Comparing the fitted values of both models seemingly is in favor of the opulent polynomial model:

```
D_4 <-
  D_4 %>%
  mutate(pred_M_6 = predict(M_6)$center,
         pred_M_7 = predict(M_7)$center)


D_4 %>%
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(aes(y = pred_M_6, color = "3rd degree polynomial")) +
  geom_smooth(aes(y = pred_M_7, color = "grand mean"))
```



But, how do both models compare in a forecasting exercise? The following generates 100 new observations, using the same generating process. The graph reveals that parsimonious model predicts the new data much better. In particular, the opulent polynomial model performs poorly for levels 1 and 2 of x_i.

```
D_5 <-
  rerun(25, sim_2(4)) %>%
  map_df(bind_rows) %>%
  left_join(select(D_4, x, starts_with("pred_")))

D_5 %>%
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(aes(y = pred_M_6, color = "3rd degree polynomial")) +
  geom_smooth(aes(y = pred_M_7, color = "grand mean"))
```



For real data sets, differences in forecasting performance are more subtle and we will need formal methods, which will be developed in the coming sections. Here, we use the root mean square error (RMSE) once again.

```
RMSE <-
  function(forecast, observed) sqrt(mean((forecast - observed)^2))

D_5 %>%
  gather(key = "Model", value = "forecast", starts_with("pred_")) %>%
  group_by(Model) %>%
  summarize(RMSE(y, forecast))
```

Model	RMSE(y, forecast)
pred_M_6	1.211
pred_M_7	0.988

```
detach(Chapter_WM)
```

The grand mean model has a much smaller RSME, which means that on average, its forecasts are closer to the new observations. How can it happen, that a simpler model has a much worse fit on face value, but predicts future data better? And the grand mean model is not just simpler than the polynomial, it even is a special case of the latter, which is also called *nested models*. If we force three of the polynomial coefficients β_1, β_2 and β_3 to zero, what remains is a flat line, $y_i = \beta_0$.

The polynomial model can take any curved path around one local minimum and one local maximum, as well as all lower-degree shapes, such as paraboles, slopes and flat lines.

In conclusion, when forcasting matters, researchers should aim for parsimonous models. In the following section, we will see how forecasting performance can be assessed without the need to gather additional data, the leave-one-out (LOO) cross-validation method. This method can be seen as the gold standard, but is very computing intensive. The subsequent section will finally introduce information criteria as computationally more efficient method to compare forecasting accuracy of models.

Leave-one-out cross validation

In the previous section we evaluated the forecasting accuracy of models by comparing retrodictions to new data. Waiting for new data before you can do model evaluation sounds awful, but this is what cross validation requires. The good news is that new data does not have to come from a new study. More precisely, cross validation only requires that the forecast data is not part of the sample you trained the model with. Psychometricians, for example, use the split half technique to assess the reliability of a test. The items of the test are split in half, one training set and one forecasting set. If the estimated participant scores correlate strongly, the test is called reliable.

So, model evaluation can be done, by selecting on part of the data to train the model, i.e. estimate the coefficients, and try to forecast the other part of the data. However, data is precious and reserving half of it for forecasting will considerably be at the cost of certainty. Fortunately, nobody actually said it has to be half the data. Another method of splitting has become common, *leave-one-out (LOO) cross validation*. The idea is simple:

1. Remove observation i from the data set.
2. Estimate the model $M_{/i}$.
3. Predict observation i with Model $M_{/i}$.
4. Measure the forecasting accuracy for i.
5. Repeat steps 1 to 4 until all observations have been left out and forecast once.

The following code implements a generic function to run LOO.

```r
do_loo <-
  function(data,
           f_fit,
           f_predict = function(fit, obs) predict(fit, newdata = obs)$center[1]
           )
    {
    model_name <- as.character(substitute(f_fit))
    leave_out <- function(obs) data %>% slice(-obs) # Quosure
    left_out  <- function(obs) data %>% slice(obs)  # Quosure
    out <-   data_frame(Obs     = 1:nrow(data),
                        Model   = model_name,
                        Sample  = Obs %>% map(leave_out), # meta function
                        OOS     = Obs %>% map(left_out), ## out-of-sample obs
                        Fitted  = Sample %>% map(f_fit)) %>%
      mutate(forecast = unlist(map2(Fitted, OOS, f_predict)))
    return(out)
  }
```

Before we put `do_loo` to use, some notes on the programming seem in order. Despite its brevity, the function is highly generic in that it can compute leave-one-out scores no matter what model you throw at it. This is mainly achieved by using advanced techniques from *functional programming*:

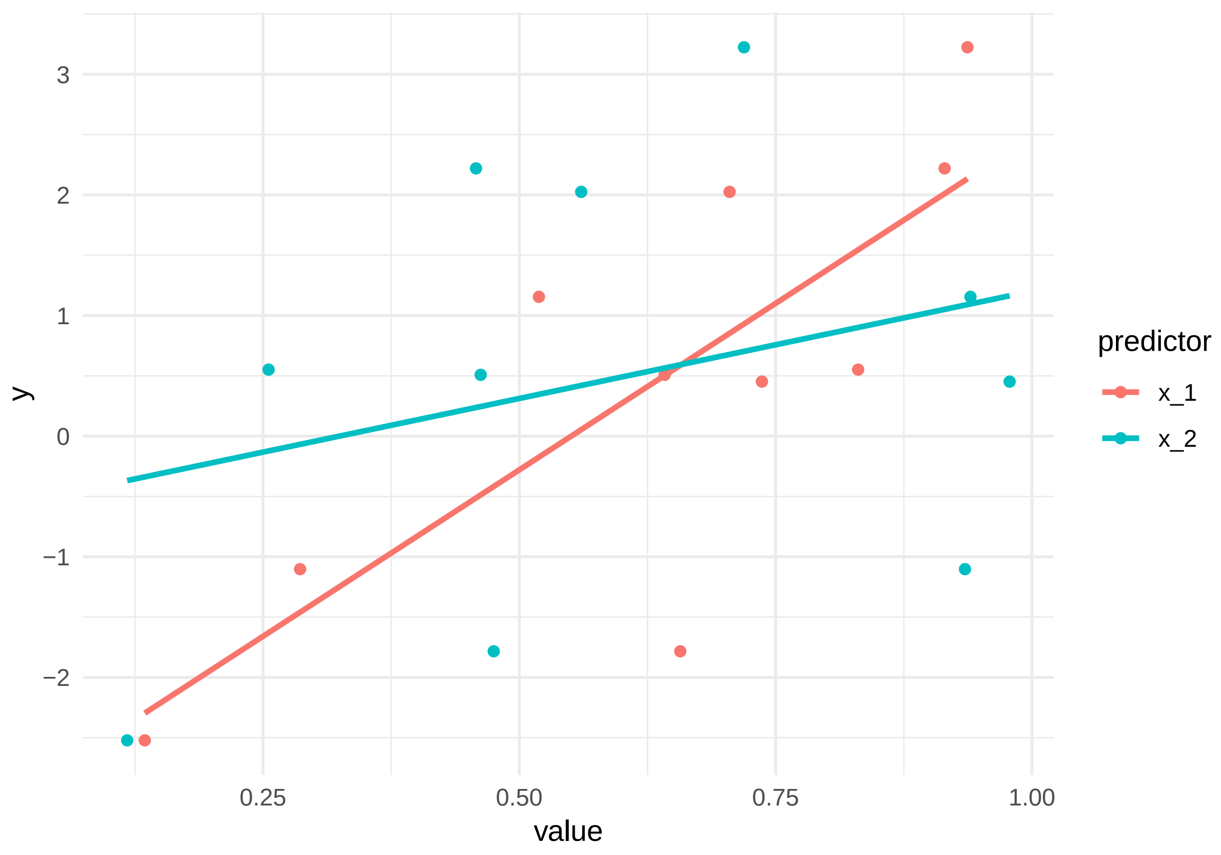
1. The argument `f_fit` takes an arbitrary function to estimate the model. This should work with all standard regression engines.

2. The argument `f_predict` takes a function as argument that produces the predicted values for the left out observations. The default is a function based on `predict` from the bayr package, but this can be accomodated.
3. The two functions that are defined inside `do_loo` are so-called *quosures*. Quosures are functions that bring their own copy of the data. They can be conceived as the functional programming counterpart to objects: Not the object brings the function, but the function brings its own data. The advantage is mostly computational as it prevents data to be copied every time the function is invoked.
4. `map` is a meta function from package purrr. It takes a list of objects and applies an arbitrary function, provided as the second argument.
5. `map2` takes two parallel input lists and applies a function. Here the forecast is created by matching observations with the model they had been excluded from.
6. The function output is created as a *tibble*, which is the tidy re-implementation of data frames. Different to original `data.frame` objects, tibbles can also store complex objects. Here, the outcome of LOO stores every single sample and estimated model, neatly aligned with its forecast value.
7. Other than one might expect, the function does not return a single score for forecasting accuracy, but a dataframe with inidividual forecasts. This is on purpose as there is more than one possible function to choose from for calculating a single accuracy score.
8. The function also makes use of what is called non-standard evaluation. This is a very advanced programming concept in R. Suffice it to say that `substitute()` captures an expression, here this is the fitting function argument, without executing it, immediatly. Here the provided argument is converted to character and put as an identifier into the dataframe. That makes it very easy to use `do_loo` for multiple models, as we will see next.

In the following illustration, we perform a LOO cross validation on a small data set `D_5`, with two predictors x_1 and x_2, where x_1 has a linear relationship with outcome y and x_2 is competely unrelated. In the following, we apply LOO to estimate three models: a grand mean model, a linear regression model on x_1 and a double linear regression with x_1 and x_2 as predictors.

```r
attach(Chapter_WM)
```

```r
sim_3 <- function(n) data_frame(Obs = 1:n,
                                x_1 = runif(n),
                                x_2 = runif(n),
                                y = rnorm(n, x_1))
set.seed(42)
D_6 <- sim_3(10)

D_6 %>%
  gather("predictor", "value", x_1:x_2) %>%
  ggplot(aes(x = value, y = y, color = predictor)) +
  geom_point() +
  geom_smooth(method = "lm", se = F)
```



Next, we define the three estimation functions and apply LOO.

```
niter = 400

fit_M_08  <-  function(sample)
  stan_glm(y ~    1,
           iter = niter, data = sample)

fit_M_09  <-  function(sample)
  stan_glm(y ~    x_1,
           iter = niter, chain = 4, data = sample)

fit_M_10  <-  function(sample)
  stan_glm(y ~    x_1 + x_2,
           iter = niter, chain = 4, data = sample)


Loo <- bind_rows(do_loo(D_6, fit_M_08),
                 do_loo(D_6, fit_M_09),
                 do_loo(D_6, fit_M_10))
```

Note, how the output of the three runs of `do_loo` are simply merged into one dataframe. This allows us to easily create a combined summary based on the root mean square error:

```
Loo %>%
  left_join(D_6, by = "Obs") %>%
  group_by(Model) %>%
  summarize(RMSE = RMSE(forecast, y))
```

Model	RMSE
fit_M_08	1.94
fit_M_09	1.28
fit_M_10	1.30

HERE

As expected, the three

```
detach(Chapter_WM)
```

Information criteria

So far, we have seen that the right level of parsimony is essential for good forecasting accuracy. While LOO can be considered gold standard for assessing forecasting accuracy, it has a severe downside. Estimating Bayesian models with MCMC is very computing intensive and for some models in this book, doing a single estimating is in the range of dozens of minutes to more than an hour. LOO estimates the model as many times as there are observations, which makes it a very inconvenient choice for more complex models and larger data sets.

Using highly oversaturated models is catastrophic for prediction. When the saturated model is asked to predict future observations, it will recover the full noise that was in the data, which is $mu_i + \epsilon_i$. Residuals represents the stochastical component, which by its very nature does not reproduce on the next occasion. Any more complex model reduces deviation from the observed data, but has less predictive value. The AIC accounts for the reduced predictive accuracy by penalizing the number of parameters.

The oldest of all IC is the *Akaike Information Criterion (AIC)*. Under a Bayesian perspective, it has limited value, but its formula will be instructive to point out how the trade-off between cling and parsimony is solved. The AIC carries one term that increases when the model moves away from the observed data points. Another term increases with the number of parameters.

$$AIC = log(p(D|\hat{\Theta})) + k$$

The AIC contains the well known likelihood, which combines the probability of all individual observations into one probability score. While Bayesians are used to think of many likelihoods, every MCMC run has their own, this likelihood is the one in our parameter space that does best, it is therefore called the *maximum likelihood estimate (MLE)* It is intuitively clear that this term decreases, when the observed values are close to the best prediction of a model.

AIC

By two simple counter-acting terms the AIC brings model fit and complexity into balance. As it purely draws principles of maximum likelihood estimation, it turns out both terms are limited to a certain class of models: the likelihood part by definition depends on the data, but prior knowledge does not occur. Under a Bayesian perspective one has to always use flat priors. The likelihood part also contains just the point estimates for parameters. Arguably, these are the maximum likelihood estimates, but this is still much less than the full posterior distribution. The likelihood term forbids to make full use of the Bayesian paradigm; the penality term limits the AIC to a certain class of models, namely those where all parameters are independent of each other. In particular, this is not the case with random effects. A random effect, we recall, is a factor, where the individual means mu_i are assumed to come from a shared distribution $mu_i\ N(\mu, \sigma)$. In a classic ANOVA, the group means of, say the ToT in two design conditions are allowed to vary freely. For a random factor, the individual μ_i determine $\mu and \sigma$, but there is feedback: if most observations happen to be in a close range, hence small σ, extreme observations gravitate towards the group mean. The smaller the dispersion, the stronger this pull and the less independent are these parameters and the less meaningless they get. As this is a process that happens gradually to every parameter, it cannot be solved by removing individual parameters (by model selection).

The *Deviance Information Criterion (DIC)* is a generalization of the *AIC* that solves two of the above issues. With the DIC, priors can be specified [check if there are limitations, for example only normal priors], and the penalty term is adjusted for dependence between parameters. In conequence, the DIC can be used in a Bayesian analysis and covers the important class of (G)LMM. The only remaining issue is that it still draws upon point estimates, the posterior modes.

Model pruning

If one measures two predictors x_1, x_2 and one outcome variable y, formally there exist four potential classic linear models to choose from:

- `y ~ 1` (grand mean)
- `y ~ x_1` (main effect 1)
- `y ~ x_2` (main effect 2)
- `y ~ x_1 + x_2` (both main effects)
- `y ~ x_1 * x_2` (both main effects and interaction)

For a data set with three predictors, the set of possible models is already 18. Some models can be discarded by more or less informal reasoning. For example, even though most studies routinely gather the variable gender, there is often little reason to assume that it makes a difference.

For a number of reasons, it can be appealing to have more formalized criteria for which model to prefer (and by how much):

1. wanting to prune the model to ease its computation
2. wanting to prune the model to ease its presentation (conciseness)
3. wanting to obtain the best predictions for future values
4. selecting one of two very similar predictors, e.g. two procedures to score results form the same test
5. wanting to test a theory

The uncanny valley observation suggests that participants likeablity judgements follow a third-degree polynomial, as this is the smallest polynomial to allow for the trough and the shoulder left of it. Previously, we have already confirmed that the 3-deg polynomial predicts more accurately than a straight linear regression model. However, it could still be the case that a 2-deg polynomial is sufficient, as it allows for the valley. The cubic term just adds some more flexibility in the curvature. But, is that really required?

```r
attach(Uncanny)
```

```r
M_2 <-
  UV_1 %>%
  mutate(huMech_0 = 1,
         huMech_1 = huMech,
         huMech_2 = huMech^2) %>%
stan_glm(avg_like ~ huMech_1 + huMech_2, data = .)
```

```
loo::waic(M_1)
```

```
##
## Computed from 4000 by 800 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic  -1481.9 20.2
## p_waic         1.9  0.2
## waic        2963.8 40.4
```

```
loo::waic(M_2)
```

```
##
## Computed from 4000 by 80 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic    -10.9  5.1
## p_waic         3.4  0.5
## waic          21.8 10.2
```

```
detach(Uncanny)
```

Choosing response distributions

In hypothesis tests, two (typically nested) models are compared for the purpose of deciding between theories. Model pruning proceeds from a full model and removes predictors for the sake of better certainty in predictions. Both are special cases of model selection. But, with modern evaluation criteria, there is more we can do to arrive at an optimal model.

The new Bayesian model selection criteria LOO and WAIC surpass the classic ones (F-test, AIC and DIC) in that they also allow to compare models with different response distributions. Recall the rather informal comparison of ToT distributions, Gaussian, gamma and exgaussian from @ref(exgaussian-regression). By visual exploration of residuals, we concluded that the exgaussian is most suited for ToT and RT outcomes. Can we confirm this by formal model selection criteria?

```
attach(CUE8)
```

```
brms::waic(M_4_gau)
```

```
##
## Computed from 4000 by 584 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic  -3582.6 36.9
## p_waic        85.7 10.6
## waic        7165.3 73.9
```

```
brms::waic(M_4_gam)
```

```
##
## Computed from 3000 by 584 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic  -3298.3 27.7
## p_waic        72.7  5.1
## waic        6596.7 55.4
```

```r
brms::waic(M_4_exg)
```

```
##
## Computed from 4000 by 584 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic  -3344.1 29.9
## p_waic        47.5  4.3
## waic        6688.2 59.7
```

```r
detach(CUE8)
```

```r
attach(Hugme)
```

```r
brms::waic(M_1_gau)
```

```
##
## Computed from 4000 by 4152 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic   1082.2 198.9
## p_waic        81.5  14.5
## waic       -2164.3 397.8
```

```r
brms::waic(M_1_gam)
```

```
##
## Computed from 1000 by 4152 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic   1993.0  90.9
## p_waic        82.6   6.8
## waic       -3986.0 181.8
```

```r
brms::waic(M_1_exg)
```

```
##
## Computed from 4000 by 4152 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic   2182.6  69.4
## p_waic        69.5   3.1
## waic       -4365.2 138.8
```

```r
detach(Hugme)
```

Testing hypothesis

New statistics, as understood in this book, ground on quantification of effects with accompanying levels of uncertainty.

The prevalent perspective in this book is the quantification of influence factors in the human-design encounter. In applied domains, where stakes and trade-offs are plenty, the most logical question to asked is on effect size, or: would it be worth to act on it? Applied research aims at practice by telling the impact.

If practice-oriented research is the right hand in applied sciences, the left hand is the body of theories that *explain* phenomena in the human-design encounter. And this is what most classical statistics thinking circles around: the idea of evaluating theories by testing hypothesis. Hypothesis tests are not quantitative, they are categorial. Every effect undergoing an hypothesis test can either be accepted as "significant" or be rejected. We may speak of magnitudes and uncertainties to continuous degrees, but testing a hypothesis means to define a hurdle upfront (such as the infamous $\alpha < .05$). Then, either the data jumps over it, or it does not.

Formally, all such hypothesis tests are comparisons of two models, one contains the parameter in question (the alternative model), the other one does not (the null model). The test itself estimates both models and compares them. If the null model wins, the hypothesis is rejected, if the alternative model wins, the hypothesis is accepted and indirectly, its source, the theory, gains credibility.

Recall the uncanny valley phenomenon: over a long range the emotional response to artificial faces gets more positive when faces get more humanlike. But, when the appearance of faces are getting really close to real, it drops – so the theory.

In chapter @ref(polynomial_regression) the non-linear relationship between human likeness of robot faces and emotional response was modelled as a 3-deg polynomial. We had taken the uncanny valley at face value, but now it is time to test it. Just by visual inspection of the graph below, it seems equally possible, that the relationship follows a much simpler trajectory, such as a straight regression line (M_0). As we know, such a model would have only two parameters, slope and intercept. Formally, that is just a 1-deg polynomial and we can regard it as a special case of the 3-deg polynomial model (M_1), where the quadratic and the cubic term are fixed to zero. Fixing a parameter is nothing else but creating a more parsimonous model.

$$M_1 : \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 M_0 : \beta_0 + \beta_1 x$$

Should we find, that the more parsimonous model wins the comparison, we had to take a critical stance on Mori's theory of the uncanny valley. We estimate the linear regression model M_0 and compare it to the polynomial model:

```r
attach(Uncanny)
```

```r
M_0 <- stan_glm(avg_like ~ huMech,
                data = UV_1)
```

```r
brms::waic(M_0)
```

```
##
## Computed from 4000 by 80 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic    -10.5  5.2
## p_waic         2.7  0.4
## waic          21.0 10.5
```

```r
brms::waic(M_1)
```

```
##
## Computed from 4000 by 800 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic  -1481.9 20.2
## p_waic         1.9  0.2
## waic        2963.8 40.4
```

```r
detach(Uncanny)
```

Model selection

The broadest application of model comparison arises, when a researcher just seeks to identify the one model with optimal predictive accuracy. In contrast to hypothesis testing and model pruning, this can take a fully optimistic route, without being hampered by prior expectations.

Exercises:

1. When testing the Unvcanny Valley hypothesis, we fixed two parameters at once, the quadratic and the cubic term. While the quadratic term is essential for creating the valley, the cubic term just adds flexibility to the curvature. Make an attempt of model pruning.

Literature

- MacElreaths statistical distributions
- XY chapter on prior specification
- MacElreaths information criteria
- W&L on Bayes factor and model selection
- Stan Manual