Columbia University in the City of New York

Lecture 7: Modeling I

Data Science for Business Analytics

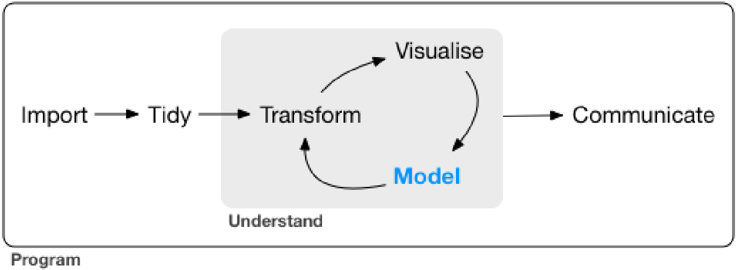
Thibault Vatter

Department of Statistics, Columbia University and HEC Lausanne, UNIL

4/9/2018

Today

Columbia University in the City of New York



- This morning:
  - how models work mechanistically (focus on linear models),
  - how to use models to find patterns in real data.
- This afternoon:
  - how to use **many** simple models,
  - how to combine modeling and programming tools.

As usual, most of the material is borrowed from R for data science.

Outline

Hypothesis generation/confirmation

1. Each observation can either be used for exploration **OR** confirmation, not both.
2. You can use an observation
   - as many times as you like for exploration,
   - only once for confirmation.

When using an observation twice, switch from confirmation to exploration.

Models

Goals:

- Provide a simple low-dimensional summary of a dataset.
- Often partition data into patterns and residuals.
- Help peel back layers of structure (since strong patterns hide subtler trends).

Two parts to a model:

1. **Family of models**: a precise, but generic, pattern to capture.
   - A straight line, or a quadratic curve.
   - Equations like `y = a_1 * x + a_2` or `y = a_1 * x ^ a_2` (with `x` and `y` known variables and `a_1` and `a_2` parameters).
2. **Fitted model**: member of the family that is closest to the data.
   - `y = 3 * x + 7` or `y = 9 * x ^ 2`.

Word of caution

All models are wrong, but some are useful. —George Box

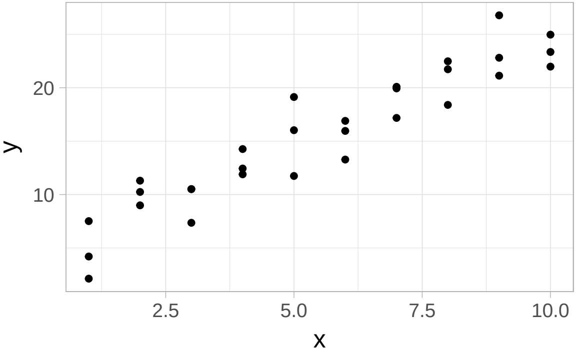
- A fitted model is "just" the closest model to the data from a family of models.
- The "best" model (according to some criteria):
  - isn't necessarily a good model,
  - isn't necessarily "true".

The goal is not to uncover truth, but to discover useful approximations.

```
library(tidyverse)
library(modelr)
```

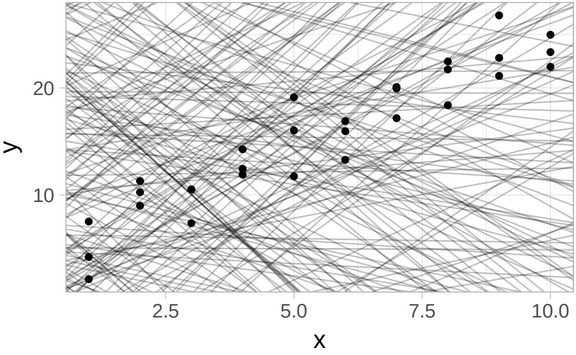
A simulated dataset


```
ggplot(sim1, aes(x, y)) + geom_point(size = 2)
```

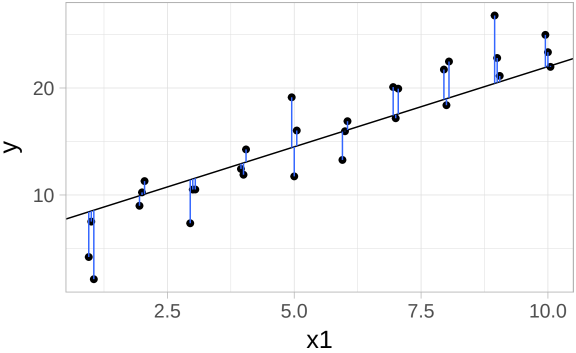


Linear family?

```
models <- tibble(a1 = runif(250, -20, 40),
                 a2 = runif(250, -5, 5))
ggplot(sim1, aes(x, y)) + geom_point(size = 2) +
  geom_abline(aes(intercept = a1, slope = a2),
              data = models, alpha = 1/4)
```



Distance between data and model



This distance is the difference between

- the y value given by the model (the **prediction**),
- and the actual y value in the data (the **response**).

Model family and RMSE

The model family:

```
model1 <- function(a, data) a[1] + data$x * a[2]

model1(c(7, 1.5), sim1)
```

```
##  [1]  8.5  8.5  8.5 10.0 10.0 10.0 11.5 11.5 11.5 13.0 13.0 13.0 14.5
## [14] 14.5 14.5 16.0 16.0 16.0 17.5 17.5 17.5 19.0 19.0 19.0 20.5 20.5
## [27] 20.5 22.0 22.0 22.0
```

Root-mean-square error (RMSE):

```
measure_distance <- function(mod, data) {
  diff <- data$y - model1(mod, data)
  sqrt(mean(diff ^ 2))}

measure_distance(c(7, 1.5), sim1)
```

```
## [1] 2.665212
```

RMSE for each model

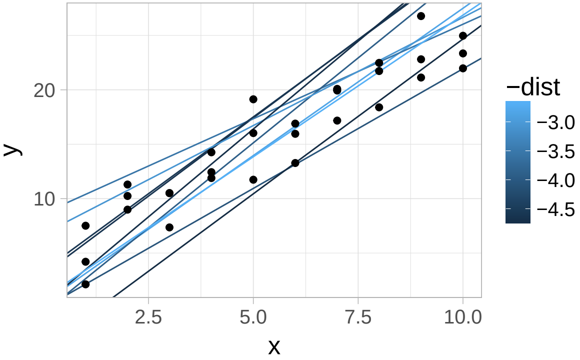
Columbia University in the City of New York

```r
sim1_dist <- function(a1, a2) measure_distance(c(a1, a2), sim1)

(models <- models %>% mutate(dist = map2_dbl(a1, a2, sim1_dist)))
```

```
## # A tibble: 250 x 3
##        a1      a2  dist
##     <dbl>   <dbl> <dbl>
##  1  33.8   4.43   43.3
##  2 - 4.07  2.62    5.82
##  3   2.33  4.33   12.7
##  4  14.4  -0.293   7.58
##  5  34.5   1.04   24.9
##  6 - 7.90 -0.150  25.1
##  7  33.9  -3.91   17.5
##  8  36.7  -2.52   15.2
##  9  19.6  -0.0149  7.50
## 10  17.7  -1.27   10.9
## # ... with 240 more rows
```

The 10 best models

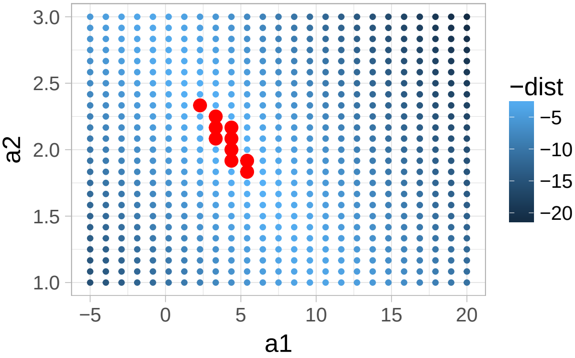

```
ggplot(sim1, aes(x, y)) +
  geom_abline(aes(intercept = a1, slope = a2, color = -dist),
              data = filter(models, rank(dist) <= 10)) +
    geom_point(size = 2)
```



Grid search

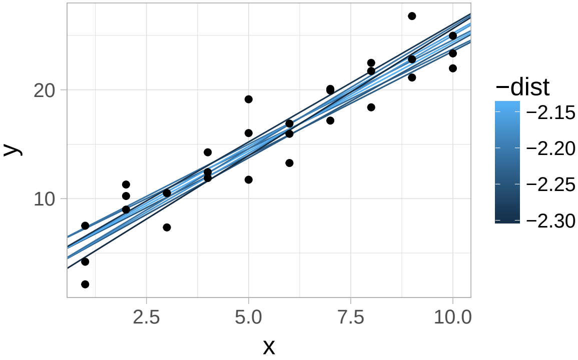
COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

```
grid <- expand.grid(a1 = seq(-5, 20, length = 25),
                    a2 = seq(1, 3, length = 25)) %>%
  mutate(dist = map2_dbl(a1, a2, sim1_dist))

ggplot(grid, aes(a1, a2)) + geom_point(aes(color = -dist)) +
  geom_point(data = filter(grid, rank(dist) <= 10),
             size = 4, color = "red")
```



Grid search cont'd


```r
ggplot(sim1, aes(x, y)) +
  geom_abline(aes(intercept = a1, slope = a2, color = -dist),
              data = filter(grid, rank(dist) <= 10)) +
    geom_point(size = 2)
```

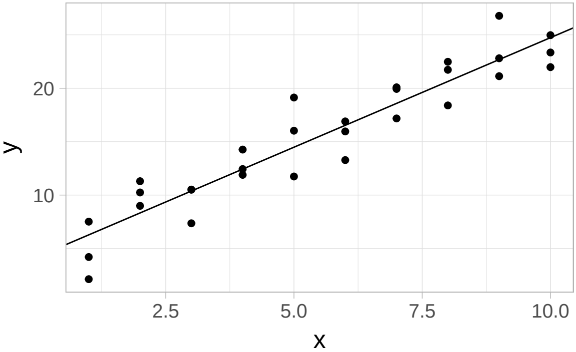


Newton-Raphson search


```r
(best <- optim(c(0, 0), measure_distance, data = sim1)$par)
```

```
## [1] 4.222248 2.051204
```

```r
ggplot(sim1, aes(x, y)) + geom_point(size = 2) +
  geom_abline(intercept = best[1], slope = best[2])
```



Linear models

- General form is e.g. `y = a_1 + a_2 * x_1 + a_3 * x_2 + ... + a_n * x_(n - 1)`
- `lm()` specify the model family using formulas (e.g., `y ~ x`, which `lm()` translates to a function like `y = a_1 + a_2 * x`)

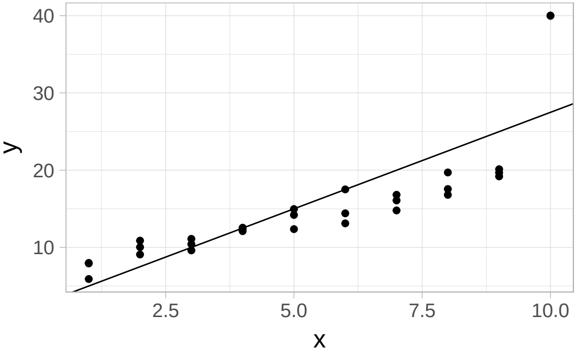
```
sim1_mod <- lm(y ~ x, data = sim1)
coef(sim1_mod)
```

```
## (Intercept)           x
##    4.220822    2.051533
```

What is going on here ?

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

```r
sim1a <- tibble(x = rep(1:10, each = 3),
                y = 6 + x * 1.5 + rnorm(length(x)))
sim1a$y[sim1a$x == max(sim1a$x)] <- 40

sim1a_mod <- lm(y ~ x, data = sim1a)
ggplot(sim1a, aes(x, y)) + geom_point(size = 2) +
  geom_abline(intercept = coef(sim1a_mod)[1], slope = coef(sim1a_mod)[2])
```



Visualizing models

Two interesting quantities to look at:

- predictions,
- and residuals.

Visualizing predictions: step 1/3


```r
(grid <- sim1 %>% modelr::data_grid(x))
```

```
## # A tibble: 10 x 1
##        x
##    <int>
##  1     1
##  2     2
##  3     3
##  4     4
##  5     5
##  6     6
##  7     7
##  8     8
##  9     9
## 10    10
```

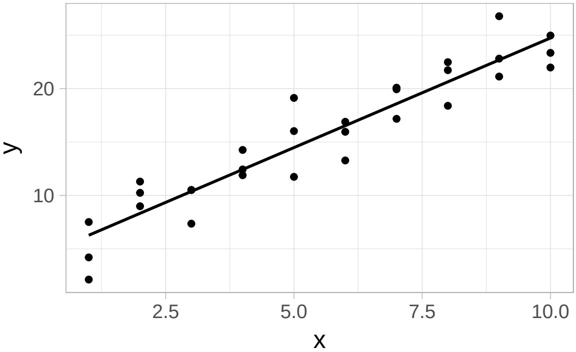
Visualizing predictions: step 2/3

```
(grid <- grid %>% add_predictions(sim1_mod))
```

```
## # A tibble: 10 x 2
##        x  pred
##    <int> <dbl>
##  1     1  6.27
##  2     2  8.32
##  3     3 10.4
##  4     4 12.4
##  5     5 14.5
##  6     6 16.5
##  7     7 18.6
##  8     8 20.6
##  9     9 22.7
## 10    10 24.7
```

Visualizing predictions: step 3/3

```r
ggplot(sim1, aes(x, y)) + geom_point(size = 2) +
  geom_line(aes(y = pred), data = grid, size = 1)
```



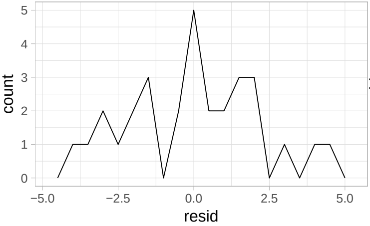
Visualizing residuals: step 1/2

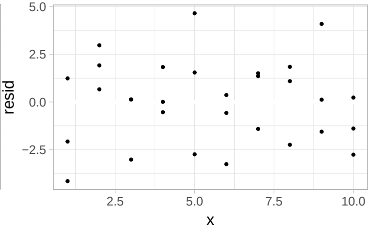
```
(sim1 <- sim1 %>% add_residuals(sim1_mod))
```

```
## # A tibble: 30 x 3
##         x     y    resid
##     <int> <dbl>    <dbl>
##  1      1  4.20 -2.07
##  2      1  7.51  1.24
##  3      1  2.13 -4.15
##  4      2  8.99  0.665
##  5      2 10.2   1.92
##  6      2 11.3   2.97
##  7      3  7.36 -3.02
##  8      3 10.5   0.130
##  9      3 10.5   0.136
## 10      4 12.4   0.00763
## # ... with 20 more rows
```

Visualizing residuals: step 2/2


```
ggplot(sim1, aes(resid)) + geom_freqpoly(binwidth = 0.5)

ggplot(sim1, aes(x, resid)) + geom_ref_line(h = 0) + geom_point()
```





Exercises

1. Instead of using `lm()` to fit a straight line, you can use `loess()` to fit a smooth curve. Repeat the process of model fitting, grid generation, predictions, and visualisation on `sim1` using `loess()` instead of `lm()`. How does the result compare to `geom_smooth()`?
2. `add_predictions()` is paired with `gather_predictions()` and `spread_predictions()`. How do these three functions differ?
3. What does `geom_ref_line()` do? What package does it come from? Why is displaying a reference line in plots showing residuals useful and important?
4. Why might you want to look at a frequency polygon of absolute residuals? What are the pros and cons compared to looking at the raw residuals?

Formulas

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

- A way of getting “special behavior”.
- “Capture variables” so they can be interpreted by the function.
- Sometimes called “Wilkinson-Rogers notation” from Symbolic Description of Factorial Models for Analysis of Variance

Behind the scenes:

```
df <- tribble(~y, ~x1, ~x2,
              4, 2, 5,
              5, 1, 6)
model_matrix(df, y ~ x1)
```

```
## # A tibble: 2 x 2
##   `(Intercept)`    x1
##           <dbl> <dbl>
## 1          1.00  2.00
## 2          1.00  1.00
```

Formulas cont'd

Without intercept:

```
model_matrix(df, y ~ x1 - 1)
```

```
## # A tibble: 2 x 1
##      x1
##   <dbl>
## 1  2.00
## 2  1.00
```

Adding a second variable:

```
model_matrix(df, y ~ x1 + x2)
```

```
## # A tibble: 2 x 3
##   `(Intercept)`    x1    x2
##           <dbl> <dbl> <dbl>
## 1          1.00  2.00  5.00
## 2          1.00  1.00  6.00
```

Categorical variables

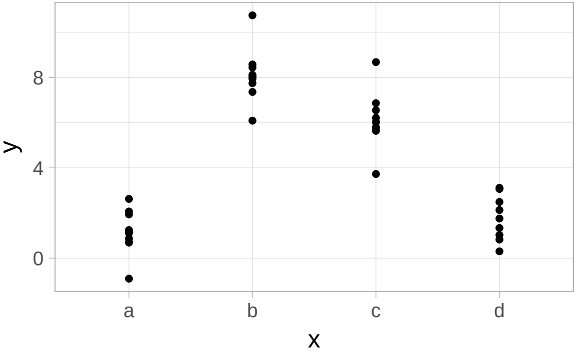
```
df <- tribble(~ sex, ~ response,
              "male", 1,
              "female", 2,
              "male", 1)
model_matrix(df, response ~ sex)
```

```
## # A tibble: 3 x 2
##   `(Intercept)` sexmale
##           <dbl>   <dbl>
## 1          1.00    1.00
## 2          1.00    0
## 3          1.00    1.00
```

Why doesn't R also create a `sexfemale` column?

Another simulated dataset


```r
ggplot(sim2, aes(x, y)) + geom_point(size = 2)
```



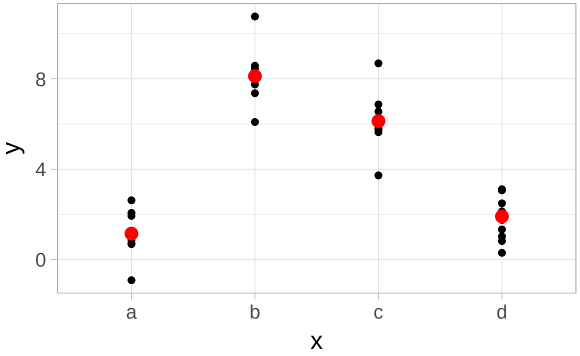
Linear model and predictions


```
mod2 <- lm(y ~ x, data = sim2)

(grid <- sim2 %>% data_grid(x) %>% add_predictions(mod2))
```

```
## # A tibble: 4 x 2
##   x      pred
##   <chr> <dbl>
## 1 a      1.15
## 2 b      8.12
## 3 c      6.13
## 4 d      1.91
```

Visualize the results


```
ggplot(sim2, aes(x)) + geom_point(aes(y = y), size = 2) +
  geom_point(data = grid, aes(y = pred), color = "red", size = 4)
```



What's happening here?

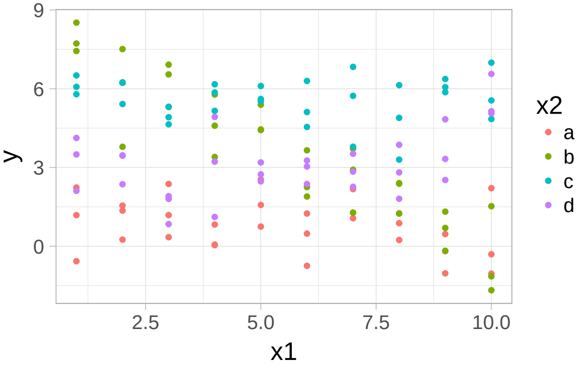
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

```
tibble(x = "e") %>% add_predictions(mod2)
```

```
## Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = o
```

Interactions (cont. and cat.)


```
ggplot(sim3, aes(x1, y)) + geom_point(aes(color = x2))
```



Two possible models


```
mod1 <- lm(y ~ x1 + x2, data = sim3)
mod2 <- lm(y ~ x1 * x2, data = sim3)
```

Note that:

- `y ~ x1 + x2` is translated to `y = a_0 + a_1 * x1 + a_2 * x2`.
- `y ~ x1 * x2` is translated to `y = a_0 + a_1 * x1 + a_2 * x2 + a_12 * x1 * x2`.

Two new tricks to visualize them

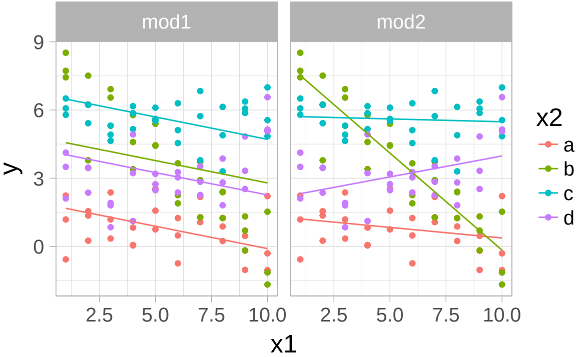
1. Give `data_grid()` both variables.
2. To generate predictions from both models simultaneously, use
   - `gather_predictions()` to add predictions as rows,
   - or `spread_predictions()` to add predictions as columns.

```
(grid <- sim3 %>% data_grid(x1, x2) %>% gather_predictions(mod1, mod2))
```

```
## # A tibble: 80 x 4
##    model    x1 x2     pred
##    <chr> <int> <fct> <dbl>
##  1 mod1      1 a      1.67
##  2 mod1      1 b      4.56
##  3 mod1      1 c      6.48
##  4 mod1      1 d      4.03
##  5 mod1      2 a      1.48
##  6 mod1      2 b      4.37
##  7 mod1      2 c      6.28
##  8 mod1      2 d      3.84
##  9 mod1      3 a      1.28
## 10 mod1      3 b      4.17
## # ... with 70 more rows
```

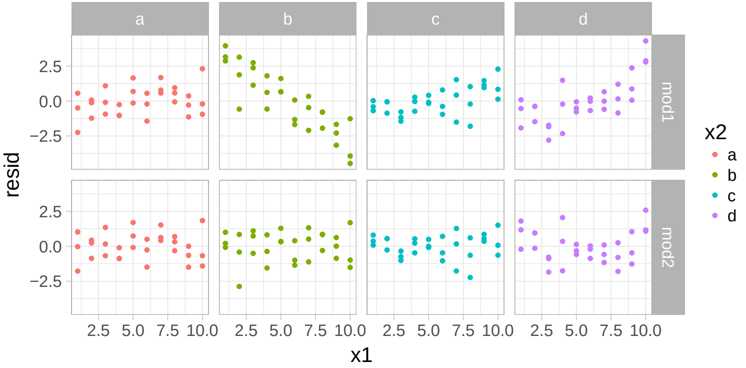
Using facetting


```
ggplot(sim3, aes(x1, y, color = x2)) + geom_point() +
  geom_line(data = grid, aes(y = pred)) + facet_wrap(~ model)
```



Which model is better?


```
sim3 <- sim3 %>% gather_residuals(mod1, mod2)

ggplot(sim3, aes(x1, resid, color = x2)) + geom_point() +
  facet_grid(model ~ x2)
```



Interactions (two continuous)

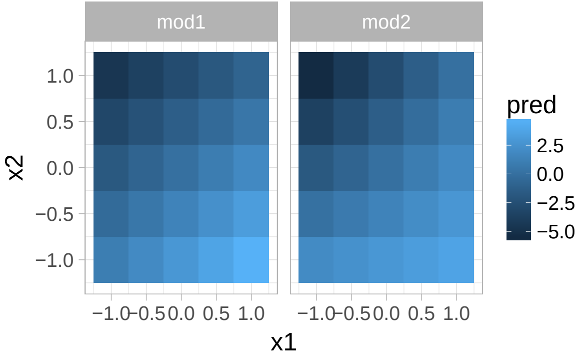

```r
mod1 <- lm(y ~ x1 + x2, data = sim4)
mod2 <- lm(y ~ x1 * x2, data = sim4)

(grid <- sim4 %>%
        data_grid(x1 = seq_range(x1, 5), x2 = seq_range(x2, 5)) %>%
        gather_predictions(mod1, mod2))
```

```
## # A tibble: 50 x 4
##    model     x1     x2   pred
##    <chr>  <dbl>  <dbl>  <dbl>
##  1 mod1  -1.00  -1.00   0.996
##  2 mod1  -1.00  -0.500 -0.395
##  3 mod1  -1.00   0     -1.79
##  4 mod1  -1.00   0.500 -3.18
##  5 mod1  -1.00   1.00  -4.57
##  6 mod1  -0.500 -1.00   1.91
##  7 mod1  -0.500 -0.500  0.516
##  8 mod1  -0.500  0     -0.875
##  9 mod1  -0.500  0.500 -2.27
## 10 mod1  -0.500  1.00  -3.66
## # ... with 40 more rows
```

`?seq_range` for other arguments (e.g., `pretty = TRUE` for tables).

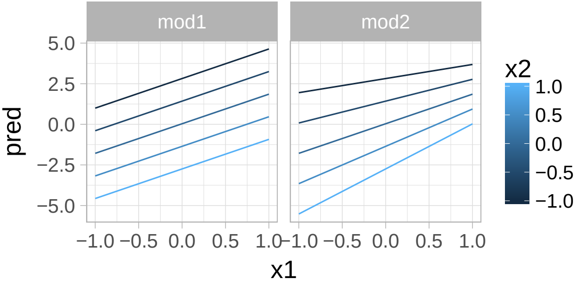
Visualize

```
ggplot(grid, aes(x1, x2)) + geom_tile(aes(fill = pred)) +
  facet_wrap(~ model)
```



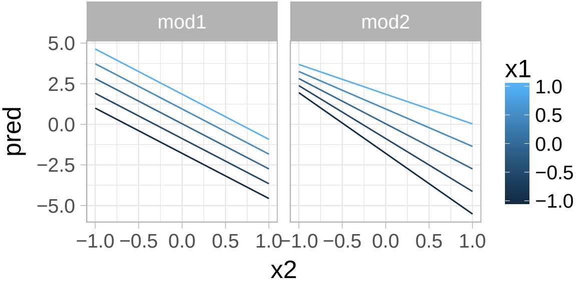
Slices with respect to x2

```
ggplot(grid, aes(x1, pred, color = x2, group = x2)) + geom_line() +
  facet_wrap(~ model)
```



Slices with respect to x1

```r
ggplot(grid, aes(x2, pred, color = x1, group = x1)) + geom_line() +
  facet_wrap(~ model)
```



Transformations

- `log(y) ~ sqrt(x1) + x2` is transformed to `log(y) = a_1 + a_2 * sqrt(x1) + a_3 * x2`.
- If the transformation involves `+`, `*`, `^`, or `-`, wrap it in `I()`:
  - `y ~ x + I(x ^ 2)` $\equiv$ `y = a_1 + a_2 * x + a_3 * x^2`.
  - `y ~ x ^ 2 + x` $\equiv$ `y ~ x * x + x` $\equiv$ `y = a_1 + a_2 * x`.

```
df <- tribble(~y, ~x, 1,  1, 2,  2, 3,  3)
model_matrix(df, y ~ x^2 + x)
model_matrix(df, y ~ I(x^2) + x)
```

```
## # A tibble: 3 x 2
##   `(Intercept)`     x
##           <dbl> <dbl>
## 1          1.00  1.00
## 2          1.00  2.00
## 3          1.00  3.00
## # A tibble: 3 x 3
##   `(Intercept)` `I(x^2)`     x
##           <dbl>    <dbl> <dbl>
## 1          1.00     1.00  1.00
## 2          1.00     4.00  2.00
## 3          1.00     9.00  3.00
```

Polynomial approximations with `poly`

To get `y = a_1 + a_2 * x + a_3 * x^2`:

```r
model_matrix(df, y ~ poly(x, 2))
```

```
## # A tibble: 3 x 3
##   `(Intercept)`          `poly(x, 2)1` `poly(x, 2)2`
##           <dbl>                  <dbl>         <dbl>
## 1          1.00 -0.707                         0.408
## 2          1.00 -0.00000000000000000785       -0.816
## 3          1.00  0.707                         0.408
```

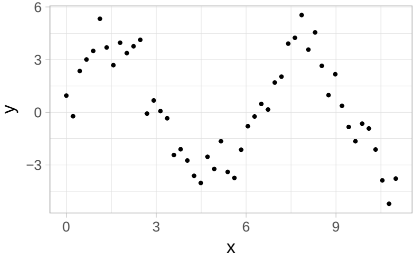
Natural splines


```
library(splines)
model_matrix(df, y ~ ns(x, 2))
```

```
## # A tibble: 3 x 3
##   `(Intercept)` `ns(x, 2)1` `ns(x, 2)2`
##           <dbl>       <dbl>       <dbl>
## 1          1.00       0           0
## 2          1.00       0.566      -0.211
## 3          1.00       0.344       0.771
```

A non-linear function


```
sim5 <- tibble(x = seq(0, 3.5 * pi, length = 50),
               y = 4 * sin(x) + rnorm(length(x)))

ggplot(sim5, aes(x, y)) + geom_point()
```

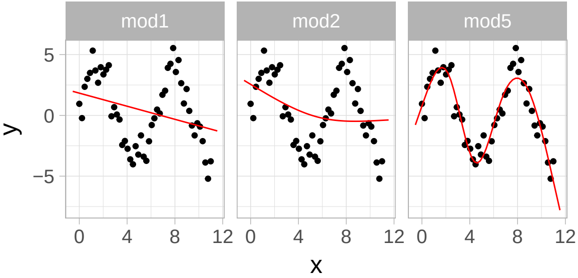


Three models using splines:

```
mod1 <- lm(y ~ ns(x, 1), data = sim5)
mod2 <- lm(y ~ ns(x, 2), data = sim5)
mod5 <- lm(y ~ ns(x, 5), data = sim5)
```

Visualize


```
grid <- sim5 %>%
  data_grid(x = seq_range(x, n = 50, expand = 0.1)) %>%
  gather_predictions(mod1, mod2, mod5, .pred = "y")

ggplot(sim5, aes(x, y)) + geom_point() +
  geom_line(data = grid, color = "red") +
  facet_wrap(~ model)
```



Missing values


```r
options(na.action = na.warn)
df <- tribble(~x, ~y,
              1, 2.2,
              2, NA,
              3, 3.5,
              4, 8.3,
              NA, 10)

mod <- lm(y ~ x, data = df)
```

```
## Warning: Dropping 2 rows with missing values
```

```r
mod <- lm(y ~ x, data = df, na.action = na.exclude)
nobs(mod)
```

```
## [1] 3
```

Other model families

Columbia University in the City of New York

- **Generalised linear models**, e.g. `stats::glm()`:
  - LMs assume continuous responses and Gaussian errors.
  - GLMs extend LMs to other distributions, including non-continuous responses (e.g. binary data or counts).
- **Generalised additive models**, e.g. `mgcv::gam()`:
  - Extend GLMs to incorporate arbitrary smooth functions.
  - A formula like `y ~ s(x)` becomes an equation like `y = f(x)`.
- **Penalized linear models**, e.g. `glmnet::glmnet()`:
  - Add penalties to favor simpler models.
  - "Generalize" better to new datasets.
- **Robust linear models**, e.g. `MASS:rlm()`:
  - Tweaks distance to downweight outliers.
  - Less sensitive to outliers, but sligthly worse without outliers.
- **Trees**, e.g. `rpart::rpart()`:
  - A piece-wise constant model splitting the data into small pieces.
  - Powerful when aggregated as **random forests** (e.g. `randomForest::randomForest()`) or **gradient boosting machines** (e.g. `xgboost::xgboost()`).

Outline

Model building

To partition data into pattern and residuals:

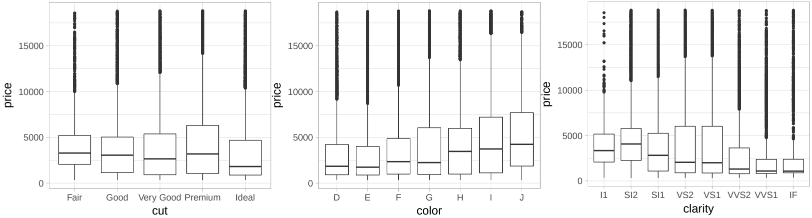
1. Find patterns with visualisation.
2. Make them concrete and precise with a model.
3. Repeat 1. and 2. after replacing the old response variable with the residuals from the model.

How about large and complex datasets?

- ML approaches “simply” focus on predictive ability.
- Issues:
  - black boxes,
  - (sometimes) hard to use domain knowledge,
  - (often) difficult to assess whether or not the model will continue to work in the long-term
- Usually, combinations of both approaches are preferred.

The diamonds dataset

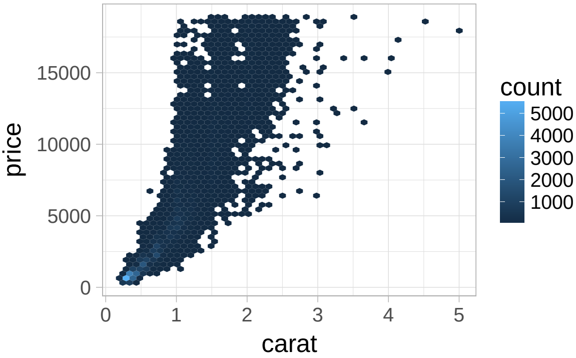
```r
ggplot(diamonds, aes(cut, price)) + geom_boxplot()
ggplot(diamonds, aes(color, price)) + geom_boxplot()
ggplot(diamonds, aes(clarity, price)) + geom_boxplot()
```



Why are low quality diamonds more expensive?

Price and carat

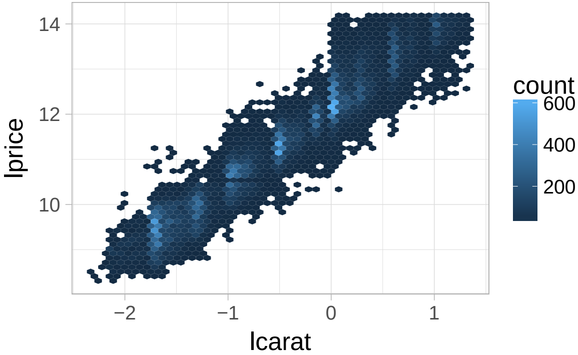

```
ggplot(diamonds, aes(carat, price)) + geom_hex(bins = 50)
```



A couple of tweaks

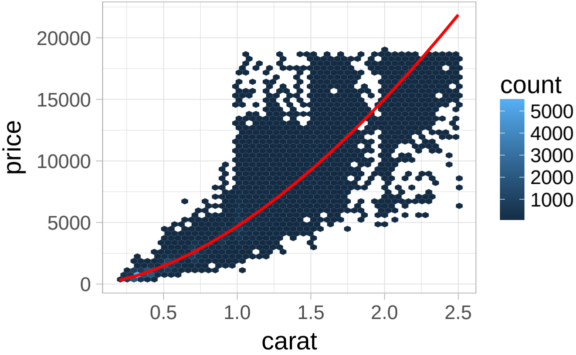
1. Focus on diamonds < 2.5 carats (99.7% of the data).
2. Log-transform the carat and price.

```
diamonds2 <- diamonds %>% filter(carat <= 2.5) %>%
  mutate(lprice = log2(price), lcarat = log2(carat))

ggplot(diamonds2, aes(lcarat, lprice)) + geom_hex(bins = 50)
mod_diamond <- lm(lprice ~ lcarat, data = diamonds2)
```



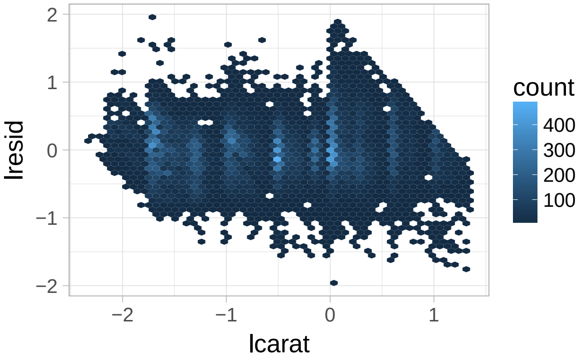
Visualize the predictions

```
grid <- diamonds2 %>% data_grid(carat = seq_range(carat, 20)) %>%
  mutate(lcarat = log2(carat)) %>%
  add_predictions(mod_diamond, "lprice") %>%
  mutate(price = 2 ^ lprice)

ggplot(diamonds2, aes(carat, price)) + geom_hex(bins = 50) +
  geom_line(data = grid, color = "red", size = 1)
```



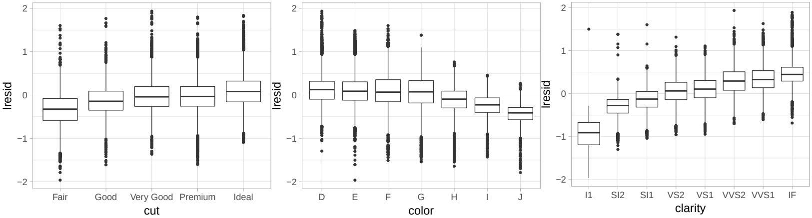
Visualize the residuals

```
diamonds2 <- diamonds2 %>% add_residuals(mod_diamond, "lresid")

ggplot(diamonds2, aes(lcarat, lresid)) + geom_hex(bins = 50)
```



Replace price by residuals


```
ggplot(diamonds2, aes(cut, lresid)) + geom_boxplot()
ggplot(diamonds2, aes(color, lresid)) + geom_boxplot()
ggplot(diamonds2, aes(clarity, lresid)) + geom_boxplot()
```



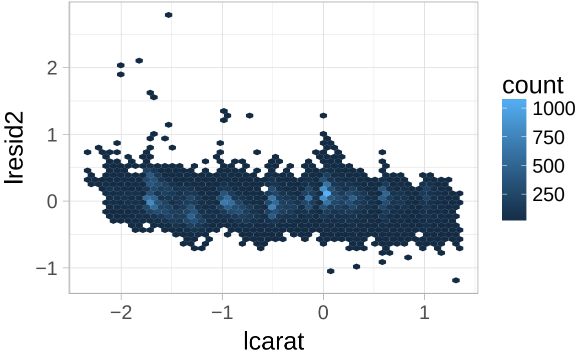
A more complicated model

```
mod_diamond2 <- lm(lprice ~ lcarat + color + cut + clarity,
                   data = diamonds2)

(grid <- diamonds2 %>%
        data_grid(cut, lcarat = -0.515,
                  color = "G", clarity = "SI1") %>%
        add_predictions(mod_diamond2))
```

```
## # A tibble: 5 x 5
##   cut       lcarat color clarity  pred
##   <ord>      <dbl> <chr> <chr>   <dbl>
## 1 Fair      -0.515 G     SI1      11.0
## 2 Good      -0.515 G     SI1      11.1
## 3 Very Good -0.515 G     SI1      11.2
## 4 Premium   -0.515 G     SI1      11.2
## 5 Ideal     -0.515 G     SI1      11.2
```

Visualize the residuals


```
diamonds2 <- diamonds2 %>%
  add_residuals(mod_diamond2, "lresid2")

ggplot(diamonds2, aes(lcarat, lresid2)) +
  geom_hex(bins = 50)
```

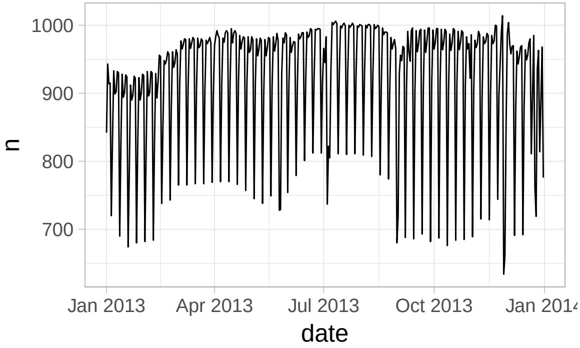


The number of daily flights

```
daily <- flights %>%
     mutate(date = make_date(year, month, day)) %>%
  group_by(date) %>%
     summarise(n = n())
```

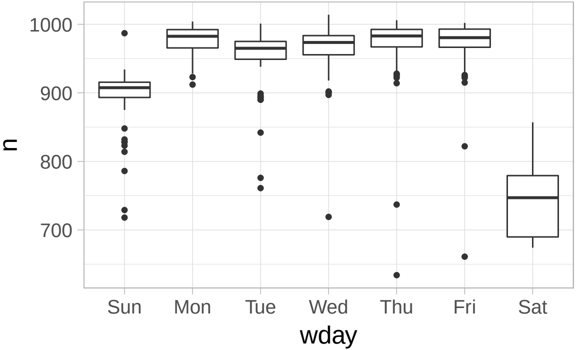
What affects this number?


```
ggplot(daily, aes(date, n)) + geom_line()
```



Day of week

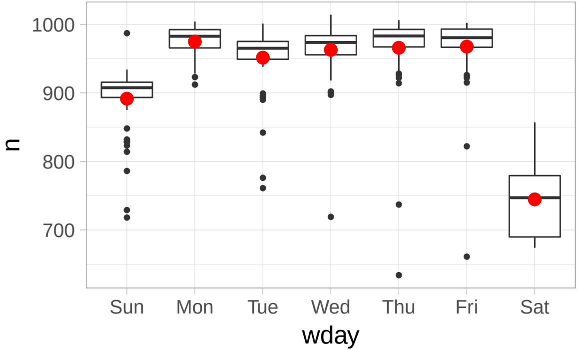

```
daily <- daily %>% mutate(wday = wday(date, label = TRUE))

ggplot(daily, aes(wday, n)) + geom_boxplot()

mod <- lm(n ~ wday, data = daily)
```



Visualize the predictions

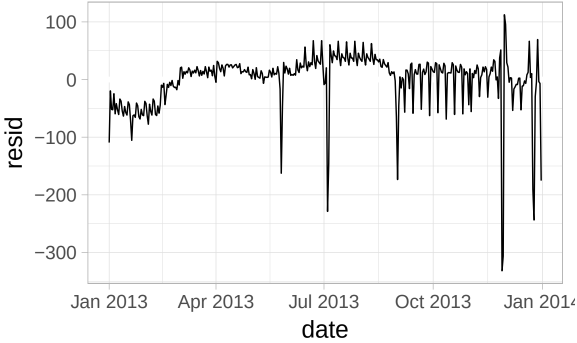
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

```
grid <- daily %>% data_grid(wday) %>%
  add_predictions(mod, "n")

ggplot(daily, aes(wday, n)) + geom_boxplot() +
  geom_point(data = grid, color = "red", size = 4)
```



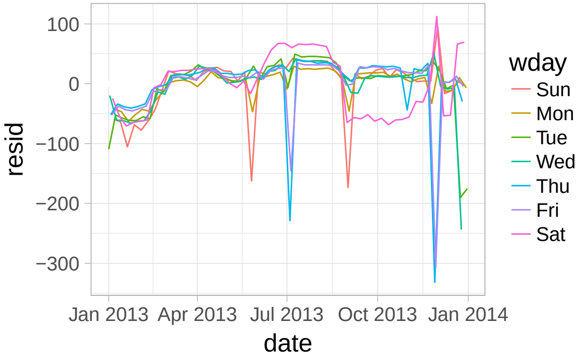
Visualize the residuals


```
daily <- daily %>% add_residuals(mod)

daily %>% ggplot(aes(date, resid)) +
  geom_ref_line(h = 0) + geom_line()
```



What happens here?

```
ggplot(daily, aes(date, resid, colour = wday)) +
    geom_ref_line(h = 0) + geom_line()
```



What happens here?

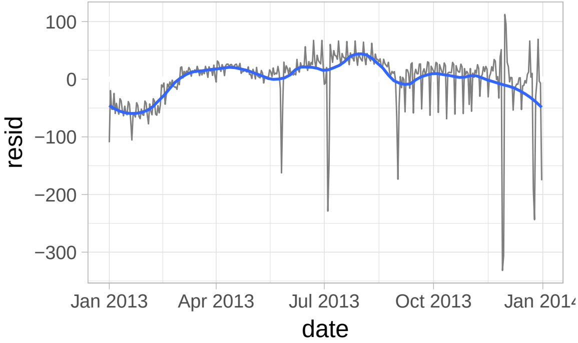
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

```r
daily %>% filter(resid < -100)
```

```
## # A tibble: 11 x 4
##    date           n wday  resid
##    <date>     <int> <ord> <dbl>
##  1 2013-01-01   842 Tue    -109
##  2 2013-01-20   786 Sun    -105
##  3 2013-05-26   729 Sun    -162
##  4 2013-07-04   737 Thu    -229
##  5 2013-07-05   822 Fri    -145
##  6 2013-09-01   718 Sun    -173
##  7 2013-11-28   634 Thu    -332
##  8 2013-11-29   661 Fri    -306
##  9 2013-12-24   761 Tue    -190
## 10 2013-12-25   719 Wed    -244
## 11 2013-12-31   776 Tue    -175
```

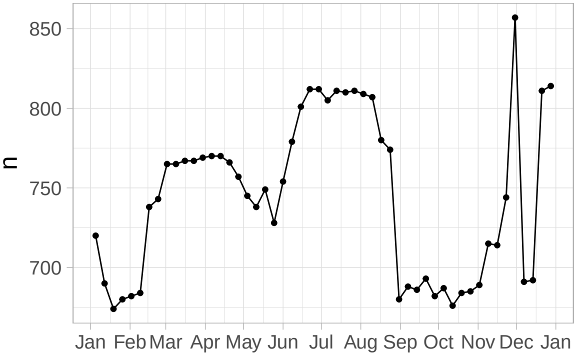
What happens here?


```
daily %>%
    ggplot(aes(date, resid)) +
    geom_ref_line(h = 0) +
    geom_line(colour = "grey50") +
    geom_smooth(se = FALSE, span = 0.20)
```



Seasonal Saturday effect

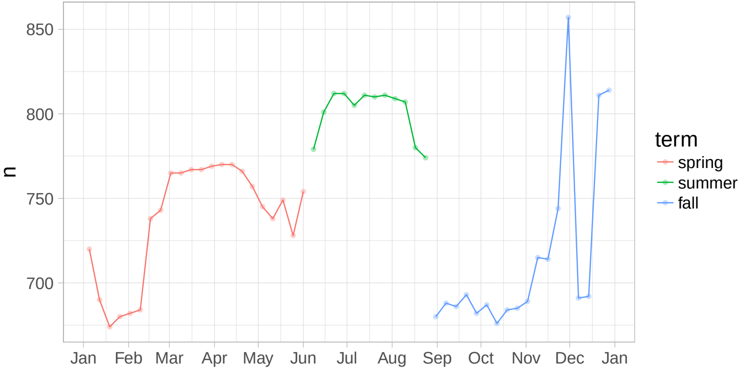
```
daily %>% filter(wday == "Sat") %>%
  ggplot(aes(date, n)) + geom_point() + geom_line() +
    scale_x_date(NULL, date_breaks = "1 month", date_labels = "%b")
```



State's school terms: summer break in 2013 was Jun 26–Sep 9.

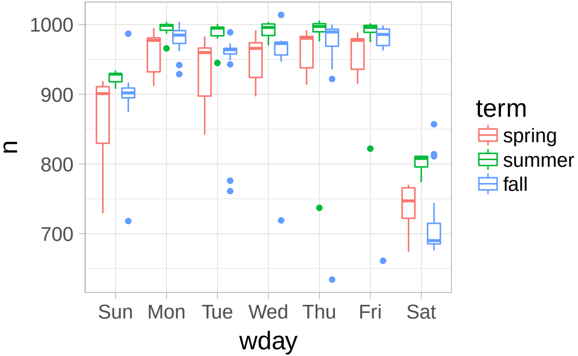
The three school terms

```
term <- function(date) {
  cut(date, breaks = ymd(20130101, 20130605, 20130825, 20140101),
      labels = c("spring", "summer", "fall"))}
daily <- daily %>% mutate(term = term(date))

daily %>% filter(wday == "Sat") %>% ggplot(aes(date, n, colour = term)) +
    geom_point(alpha = 1/3) + geom_line() +
    scale_x_date(NULL, date_breaks = "1 month", date_labels = "%b")
```



School terms and day of week

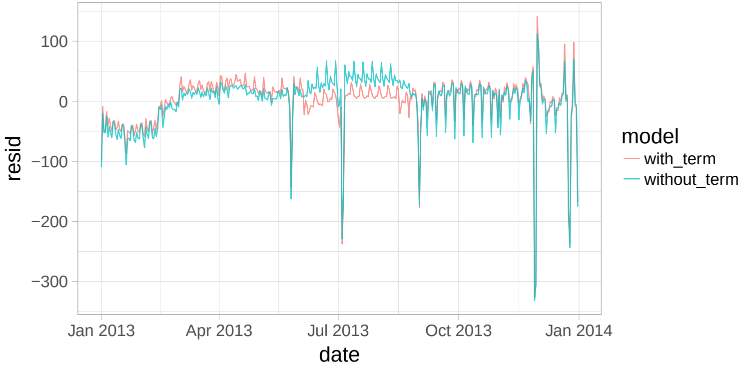

```r
daily %>%
  ggplot(aes(wday, n, colour = term)) + geom_boxplot()
```



An improved model

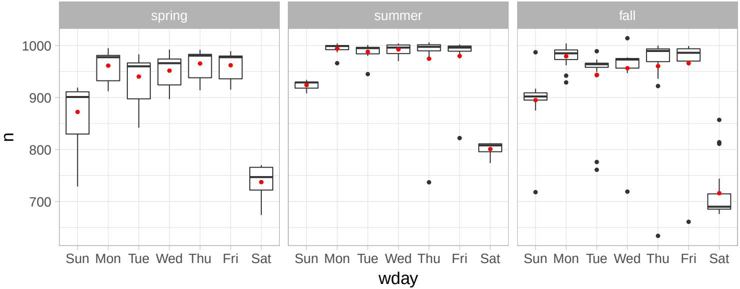
COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

```
mod1 <- lm(n ~ wday, data = daily)
mod2 <- lm(n ~ wday * term, data = daily)

daily %>% gather_residuals(without_term = mod1, with_term = mod2) %>%
  ggplot(aes(date, resid, colour = model)) + geom_line(alpha = 0.75)
```



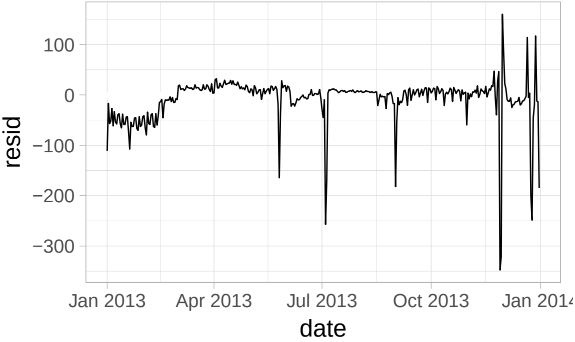
What's going on here?

```
grid <- daily %>% data_grid(wday, term) %>%
  add_predictions(mod2, "n")

ggplot(daily, aes(wday, n)) + geom_boxplot() +
  geom_point(data = grid, colour = "red") + facet_wrap(~ term)
```



Robust fit


```r
mod3 <- MASS::rlm(n ~ wday * term, data = daily)

daily %>% add_residuals(mod3, "resid") %>% ggplot(aes(date, resid)) +
  geom_hline(yintercept = 0, size = 2, colour = "white") + geom_line()
```



Computed variables

Either bundled up into a function:

```
compute_vars <- function(data) {
  data %>%
    mutate(term = term(date),
           wday = wday(date, label = TRUE))
}
```

Or directly in the model formula:

```
wday2 <- function(x) wday(x, label = TRUE)
mod3 <- lm(n ~ wday2(date) * term(date), data = daily)
```

Time of year: an alternative approach


```
library(splines)
mod <- MASS::rlm(n ~ wday * ns(date, 5), data = daily)

daily %>% data_grid(wday, date = seq_range(date, n = 13)) %>%
  add_predictions(mod) %>%
  ggplot(aes(date, pred, colour = wday)) + geom_line() + geom_point()
```