Columbia University
In the City of New York

Data Science for Business Analytics

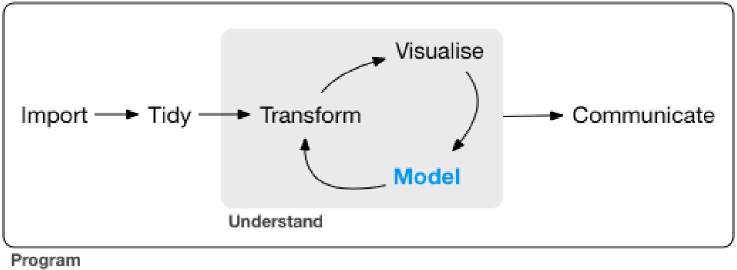
Lecture 6

Thibault Vatter

Department of Statistics, Columbia University

03/16/2020

Modeling



- First (today):
  - how models work mechanistically (focus on linear models),
  - how to use models to find patterns in real data.
- Then (next time):
  - how to use **many** simple models,
  - how to combine modeling and programming tools.
- As usual, material borrowed from R for data science.

Outline

Outline

Hypothesis generation/confirmation

- Each observation can either be used for exploration **OR** confirmation, not both.
- 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).
- The two components of a model:
  - **Family of models**: a precise, but generic, pattern to capture.
    - A straight line, or a quadratic curve.
    - Equations like `y = a1 * x + a2` or `y = a1 * x ^ a2` (with `x` and `y` known variables and `a1` and `a2` parameters).
  - **Fitted model**: member of the family that is closest to the data.
    - `y = 3 * x + 7` or `y = 9 * x ^ 2`.

Word of caution

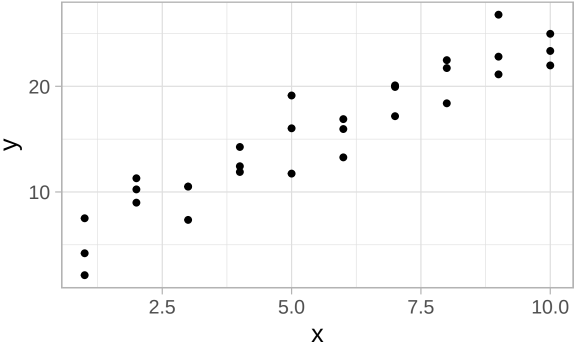
All models are wrong, but some are useful. —George Box

- 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.**

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

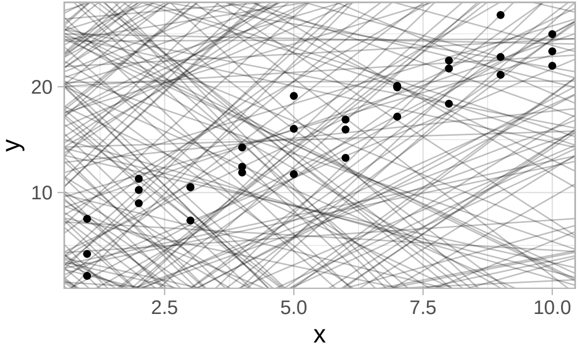
A simulated dataset


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

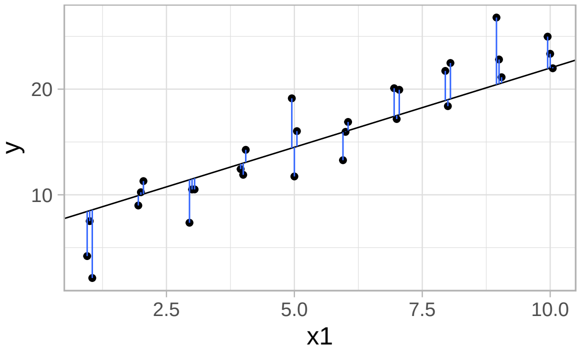


Linear family?


```
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] 2.67
```

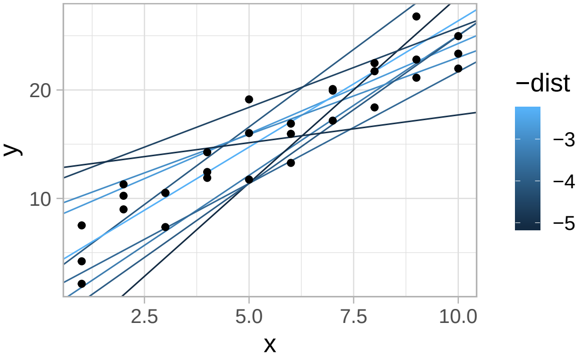
RMSE for each model

```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 -15.2   0.0889  30.8
#>  2  30.1  -0.827   13.2
#>  3  16.0   2.27    13.2
#>  4 -10.6   1.38    18.7
#>  5 -19.6  -1.04    41.8
#>  6   7.98  4.59    19.3
#>  7   9.87 -2.01    20.5
#>  8  -2.61 -4.50    46.9
#>  9  24.0   0.762   13.4
#> 10  26.4  -2.82    14.9
#> # ... with 240 more rows
```

The 10 best models

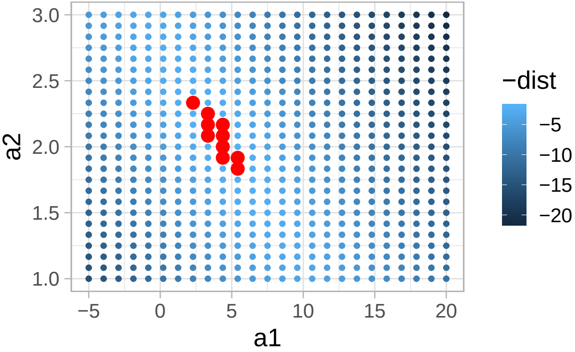
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

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



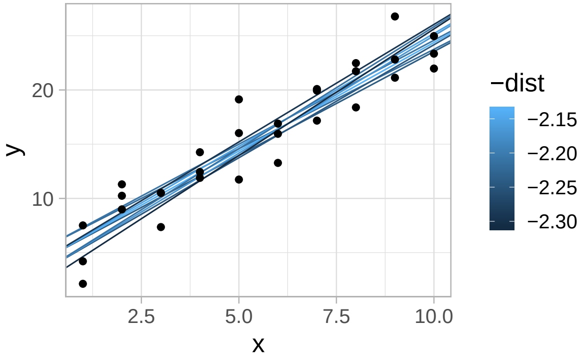
Grid search

```r
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 = grid %>% filter(rank(dist) <= 10),
             size = 4, color = "red")
```



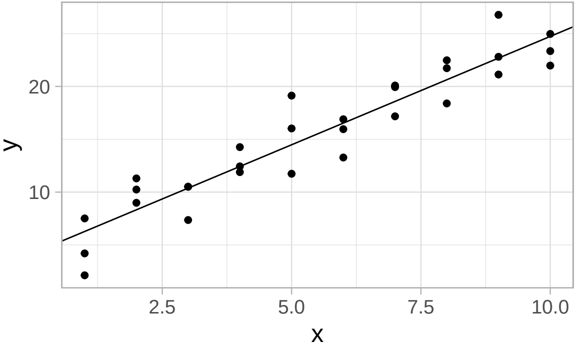
Grid search cont'd


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



Newton-Raphson search


```r
(best <- optim(c(0, 0), measure_distance, data = sim1)$par)
#> [1] 4.22 2.05

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



Linear models

- General form:
  - `y = a1 + a2 * x_1 + a3 * x_2 + ... + an * x_(n - 1)`.
- `lm()`:
  - Specify the model family using formulas!
  - E.g., `y ~ x` is translated to a function like `y = a1 + a2 * x`.

```
sim1_mod <- lm(y ~ x, data = sim1)
coef(sim1_mod)
#> (Intercept)           x
#>        4.22        2.05
```

- Two interesting quantities to look at:
  - The **predictions**.
  - The **residuals**.
- But before that...
  - The output of `lm()` in R.
  - A statistics digression.

Linear model in R

```
summary(sim1_mod)
#>
#> Call:
#> lm(formula = y ~ x, data = sim1)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -4.147 -1.520  0.133  1.467  4.652
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    4.221      0.869    4.86  4.1e-05 ***
#> x              2.052      0.140   14.65  1.2e-14 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.2 on 28 degrees of freedom
#> Multiple R-squared:  0.885,  Adjusted R-squared:  0.88
#> F-statistic:  215 on 1 and 28 DF,  p-value: 1.17e-14
```

Linear model in R cont'd


```
sim0_mod <- lm(y ~ 1, data = sim1)
anova(sim0_mod, sim1_mod)
#> Analysis of Variance Table
#>
#> Model 1: y ~ 1
#> Model 2: y ~ x
#>   Res.Df  RSS Df Sum of Sq   F  Pr(>F)
#> 1     29 1178
#> 2     28  136  1      1042 215 1.2e-14 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ... end of the digression!
- Two interesting quantities to look at:
  - The **predictions**.
  - The **residuals**.

Visualizing predictions in 3 steps

- Step 1

```
(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
```

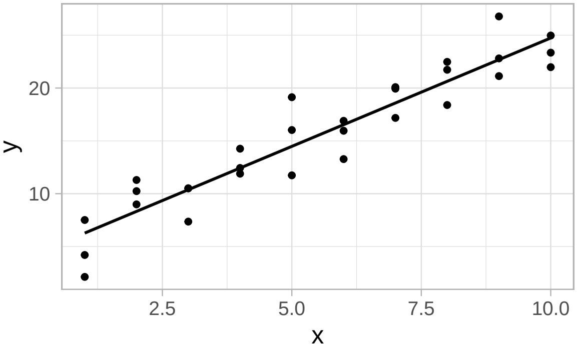
- Step 2

```
(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 in 3 steps

- Step 3

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



Visualizing residuals in three steps

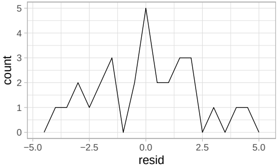
- Step 1

```
(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 in three steps

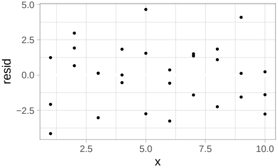
- Step 2

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



- Step 3

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



Formulas

Columbia University in the City of New York

- What's that?
  - 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     2
#> 2             1     1
```

Formulas cont'd

- Without intercept:

```
model_matrix(df, y ~ x1 - 1)
#> # A tibble: 2 x 1
#>      x1
#>   <dbl>
#> 1     2
#> 2     1
```

- Adding a second variable:

```
model_matrix(df, y ~ x1 + x2)
#> # A tibble: 2 x 3
#>   `(Intercept)`    x1    x2
#>           <dbl> <dbl> <dbl>
#> 1             1     2     5
#> 2             1     1     6
```

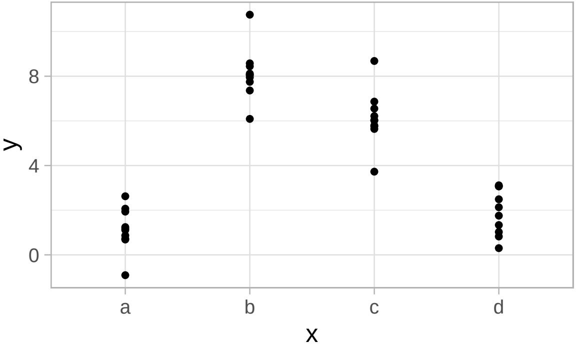
Categorical variables


```r
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       1
#> 2             1       0
#> 3             1       1
```

- 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

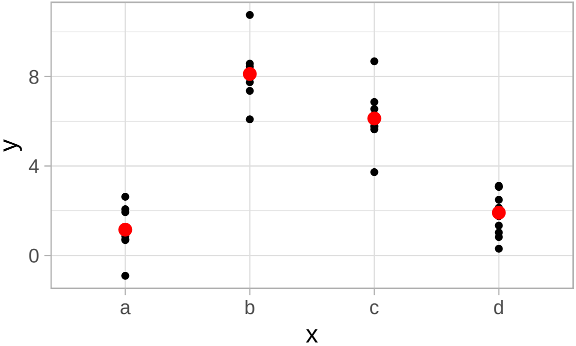
Columbia University in the City of New York

```
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

Columbia University in the City of New York

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



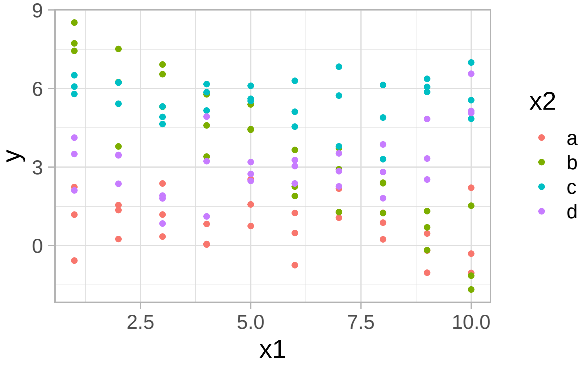
What’s happening here?

```r
tibble(x = "e") %>%
  add_predictions(mod2)
#> Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = ob
```

Interactions (cont. and cat.)

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

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



Two possible models


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

- Note that:
  - `y ~ x1 + x2` becomes `y = a0 + a1 * x1 + a2 * x2`.
  - `y ~ x1 * x2` becomes `y = a0 + a1 * x1 + a2 * x2 + a12 * x1 * x2`.

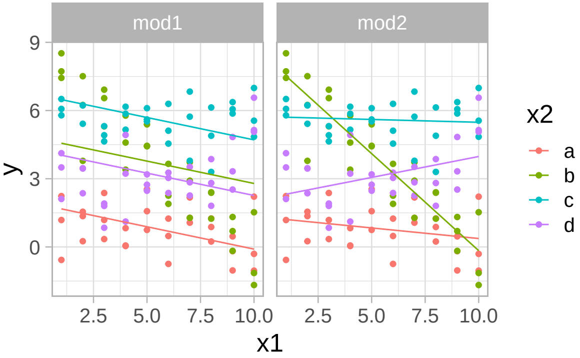
Two new tricks to visualize them

- Give `data_grid()` both variables.
- 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
```

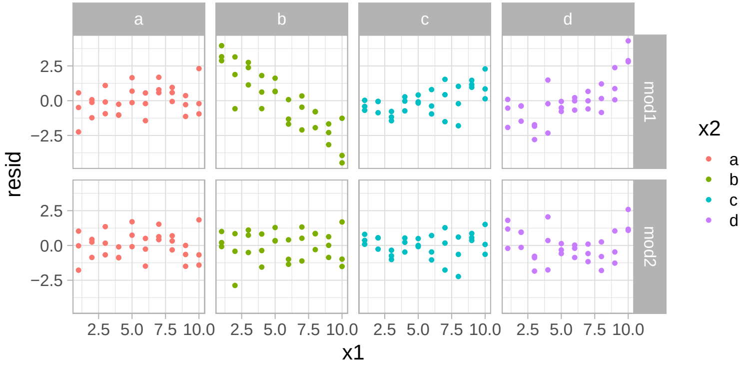
Using facetting


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



Which model is better?


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

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



Which model is better?

- Remember slide 23

```
anova(mod1, mod2)
#> Analysis of Variance Table
#>
#> Model 1: y ~ x1 + x2
#> Model 2: y ~ x1 * x2
#>   Res.Df RSS Df Sum of Sq    F Pr(>F)
#> 1    115 270
#> 2    112 118  3       153 48.5 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

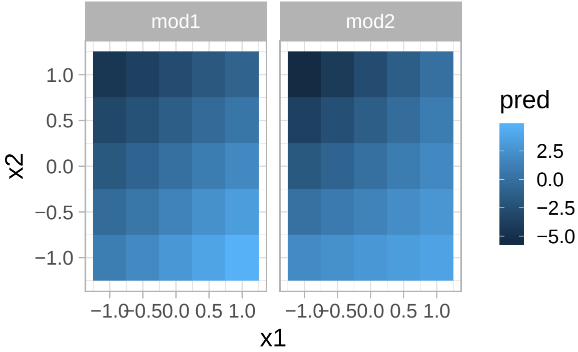
Interactions (two continuous)


```
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    -1    0.996
#>  2 mod1   -1    -0.5 -0.395
#>  3 mod1   -1     0   -1.79
#>  4 mod1   -1     0.5 -3.18
#>  5 mod1   -1     1   -4.57
#>  6 mod1   -0.5  -1    1.91
#>  7 mod1   -0.5  -0.5  0.516
#>  8 mod1   -0.5   0   -0.875
#>  9 mod1   -0.5   0.5 -2.27
#> 10 mod1   -0.5   1   -3.66
#> # ... with 40 more rows
```

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

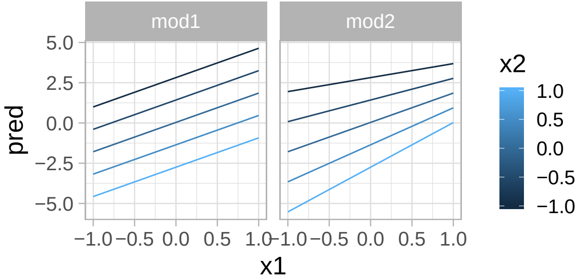
Visualize


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



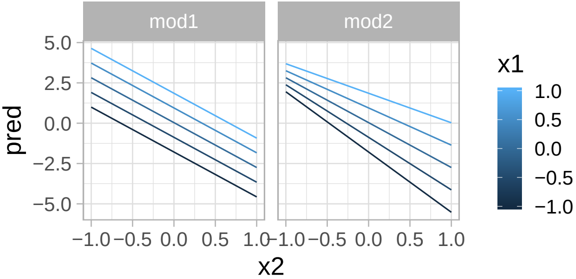
Slices with respect to x2


```r
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)
```



Which model is better?

- Remember slide 23

```
anova(mod1, mod2)
#> Analysis of Variance Table
#>
#> Model 1: y ~ x1 + x2
#> Model 2: y ~ x1 * x2
#>   Res.Df  RSS Df Sum of Sq    F Pr(>F)
#> 1    297 1323
#> 2    296 1278  1      45.2 10.5 0.0014 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Transformations

- `log(y) ~ sqrt(x1) + x2` is transformed to `log(y) = a1 + a2 * sqrt(x1) + a3 * x2`.
- If the transformation involves `+`, `*`, `^`, or `-`, wrap it in `I()`:
  - `y ~ x + I(x ^ 2)` $\equiv$ `y = a1 + a2 * x + a3 * x^2`.
  - `y ~ x ^ 2 + x` $\equiv$ `y ~ x * x + x` $\equiv$ `y = a1 + a2 * 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     1
#> 2             1     2
#> 3             1     3
#> # A tibble: 3 x 3
#>   `(Intercept)` `I(x^2)`     x
#>           <dbl>    <dbl> <dbl>
#> 1             1        1     1
#> 2             1        4     2
#> 3             1        9     3
```

- To get `y = a1 + a2 * x + a3 * x^2`:

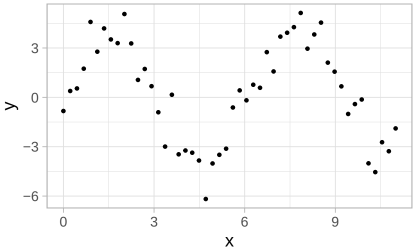
```
model_matrix(df, y ~ poly(x, 2))
#> # A tibble: 3 x 3
#>   `(Intercept)` `poly(x, 2)1` `poly(x, 2)2`
#>           <dbl>         <dbl>         <dbl>
#> 1             1      -7.07e- 1         0.408
#> 2             1      -7.85e-17        -0.816
#> 3             1       7.07e- 1         0.408
```

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       0           0
#> 2             1       0.566      -0.211
#> 3             1       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()
```

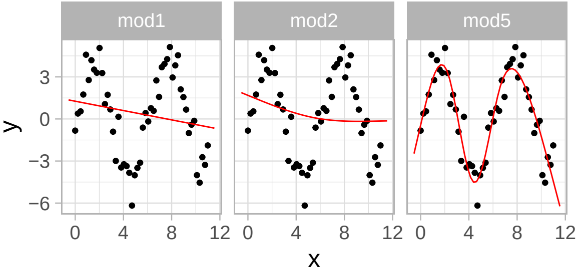


- Three models using splines:

```
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
mod <- lm(y ~ x, data = df, na.action = na.exclude)
nobs(mod)
#> [1] 3
```

Other model families

- **Generalized linear models**, e.g. `stats::glm()`:
  - While LMs assume continuous responses/Gaussian errors, GLMs extend them to other distributions, including non-continuous responses (e.g. binary data or counts).
- **Generalized additive models**, e.g. `mgcv::gam()`:
  - Extend GLMs to incorporate smooth functions
  - Formulas like `y ~ s(x)` become equations like `y = f(x)`.
- **Penalized linear models**, e.g. `glmnet::glmnet()`:
  - Add penalties to favor simpler models and "generalize" better.
- **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()`:
  - Piece-wise constant models 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

Making tidy data with broom

- `broom::glance(model)`
  - A row for each model.
  - Columns give a model summary (measure of model quality, complexity, or combination of both).
- `broom::tidy(model)`
  - A row for each coefficient in the model.
  - Columns give information about the estimate or its variability.
- `broom::augment(model, data)`
  - A row for each row in `data`.
  - Adds extra values like residuals, and influence statistics.

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm_D9 <- lm(weight ~ group)
```

broom::glance() and broom::tidy()

- broom::glance(model)
  - A row for each model.
  - Columns give a model summary.

```
broom::glance(lm_D9)
#> # A tibble: 1 x 11
#>   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC
#>       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl>
#> 1    0.0731        0.0216 0.696      1.42   0.249     2  -20.1  46.2
#> # ... with 3 more variables: BIC <dbl>, deviance <dbl>,
#> #   df.residual <int>
```

- broom::tidy(model)
  - A row for each coefficient in the model.
  - Columns give information about the estimate or its variability.

```
broom::tidy(lm_D9)
#> # A tibble: 2 x 5
#>   term        estimate std.error statistic  p.value
#>   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
#> 1 (Intercept)    5.03      0.220     22.9  9.55e-15
#> 2 groupTrt      -0.371     0.311     -1.19 2.49e- 1
```

broom::augment()

- broom::augment(model, data)
  - A row for each row in data.
  - Adds extra values like residuals, and influence statistics.

```r
broom::augment(lm_D9) %>%
  print(n = 10)
#> # A tibble: 20 x 9
#>    weight group .fitted .se.fit .resid  .hat .sigma .cooksd
#>     <dbl> <fct>   <dbl>   <dbl>  <dbl> <dbl>  <dbl>   <dbl>
#>  1   4.17 Ctl      5.03   0.220 -0.862  0.10  0.682 0.0946
#>  2   5.58 Ctl      5.03   0.220  0.548  0.10  0.703 0.0382
#>  3   5.18 Ctl      5.03   0.220  0.148  0.1   0.716 0.00279
#>  4   6.11 Ctl      5.03   0.220  1.08   0.1   0.661 0.148
#>  5   4.5  Ctl      5.03   0.220 -0.532  0.1   0.704 0.0360
#>  6   4.61 Ctl      5.03   0.220 -0.422  0.1   0.708 0.0227
#>  7   5.17 Ctl      5.03   0.220  0.138  0.1   0.716 0.00242
#>  8   4.53 Ctl      5.03   0.220 -0.502  0.1   0.705 0.0321
#>  9   5.33 Ctl      5.03   0.220  0.298  0.1   0.713 0.0113
#> 10   5.14 Ctl      5.03   0.220  0.108  0.1   0.716 0.00148
#> # ... with 10 more rows, and 1 more variable: .std.resid <dbl>
```

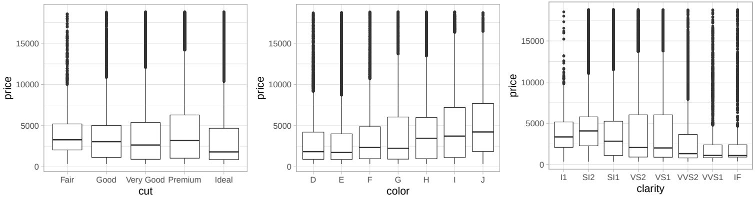
Outline

Model building

- To partition data into pattern and residuals:
  - Find patterns with visualization.
  - Make them concrete and precise with a model.
  - 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, a combination of both approaches is preferred.

The diamonds dataset

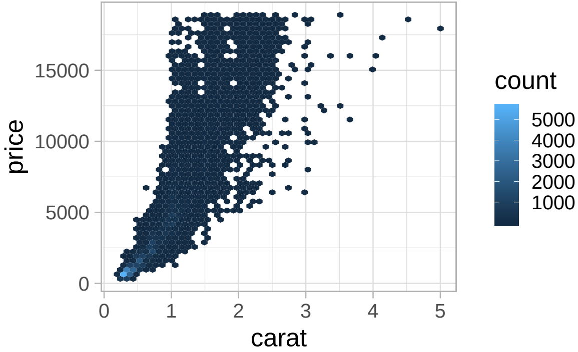
```r
ggplot(diamonds, aes(cut, price)) + geom_boxplot()
ggplot(diamonds, aes(color, price)) + geom_boxplot()
ggplot(diamonds, aes(clarity, price)) + geom_boxplot()
```



- Why are low quality diamonds more expensive?

Price and carat

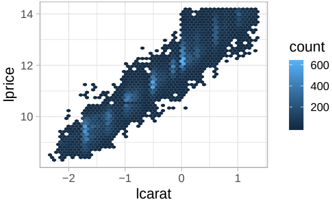

```r
ggplot(diamonds, aes(carat, price)) +
  geom_hex(bins = 50)
```



A couple of tweaks

- Focus on diamonds < 2.5 carats (99.7% of the data).
- 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)
```

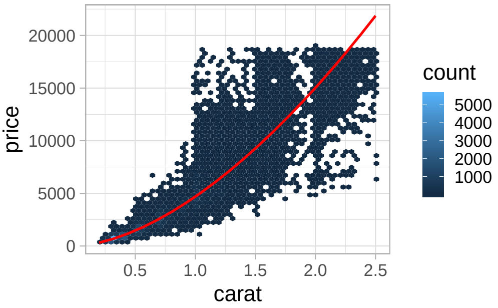


- A simple model:

```
mod_diamond <- lm(lprice ~ lcarat, data = diamonds2)
```

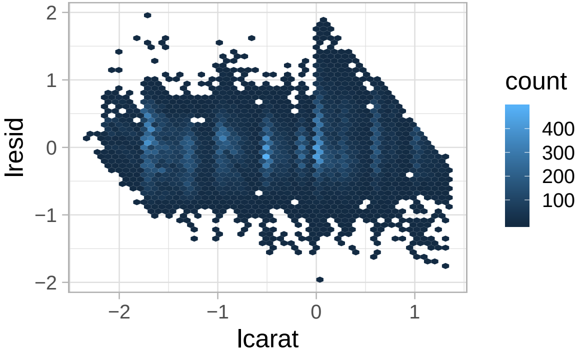
Visualize the predictions


```
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)
```



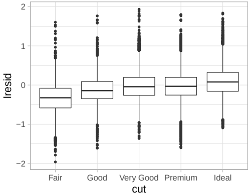
Visualize the residuals

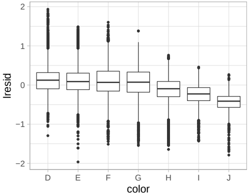

```
diamonds2 <- diamonds2 %>%
  add_residuals(mod_diamond, "lresid")

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

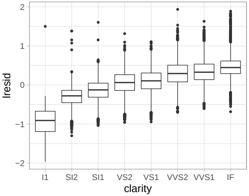


Replace price by residuals

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





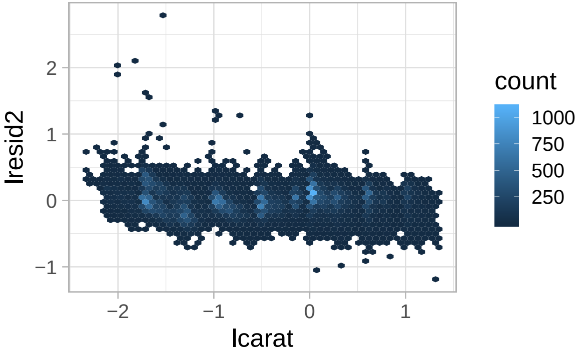


A more complicated model


```
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)
```

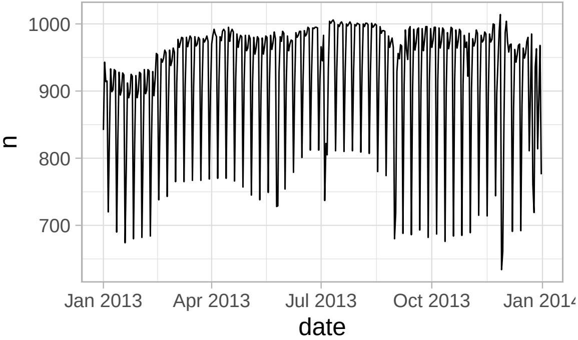


The number of daily flights


```
library(nycflights13)
library(lubridate)
daily <- flights %>%
     mutate(date = make_date(year, month, day)) %>%
  group_by(date) %>%
  summarize(n = n())
```

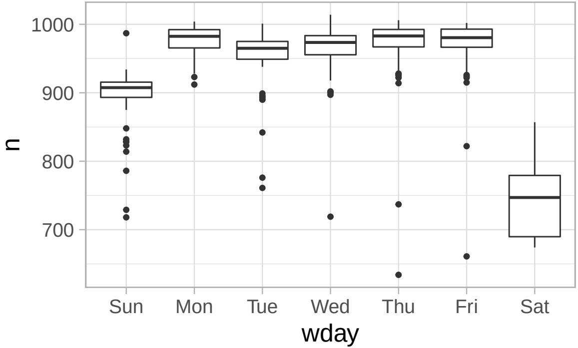
What affects this number?


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



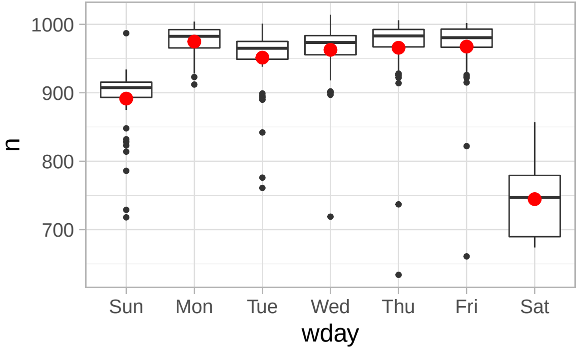
Day of week


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

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

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



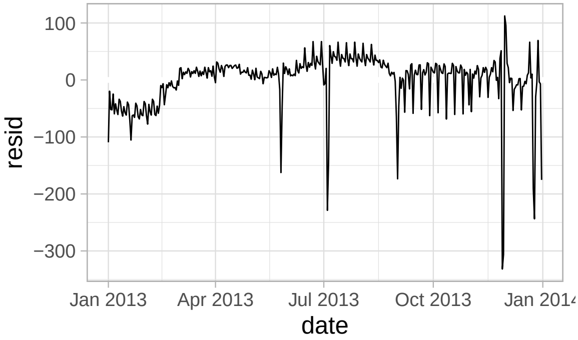
Visualize the predictions


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

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



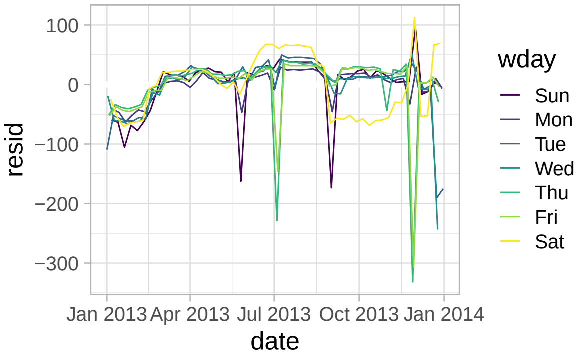
Visualize the residuals

```
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, color = wday)) +
    geom_ref_line(h = 0) + geom_line()
```



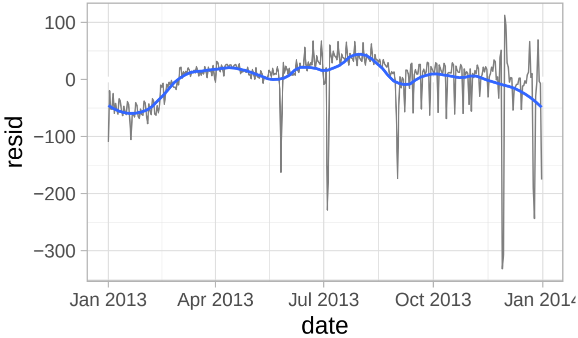
What happens here?

```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?

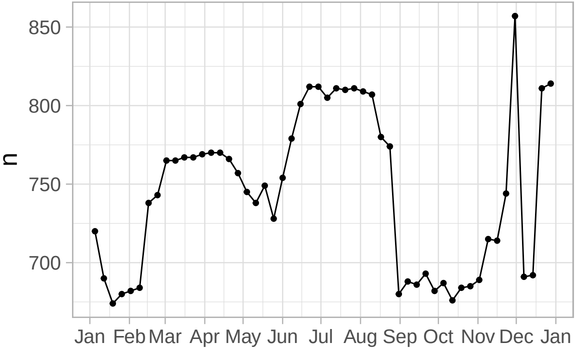
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

```r
daily %>%
    ggplot(aes(date, resid)) +
    geom_ref_line(h = 0) +
    geom_line(color = "grey50") +
    geom_smooth(se = FALSE, span = 0.20)
#> `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



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")
```



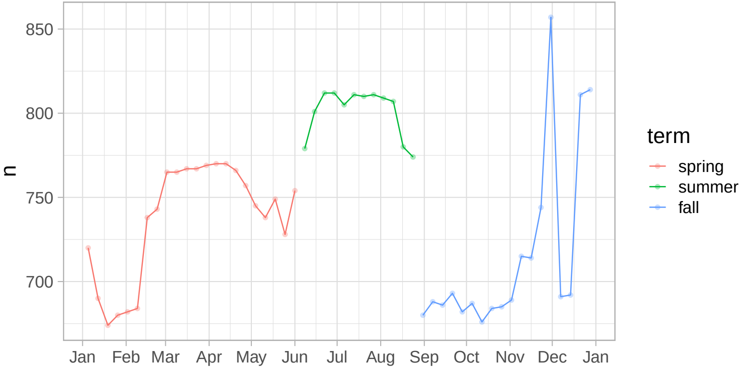
- State's school terms: summer break in 2013 was Jun 26–Sep 9.

The three school terms

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

daily
#> # A tibble: 365 x 5
#>    date           n wday   resid term
#>    <date>     <int> <ord>  <dbl> <fct>
#>  1 2013-01-01   842 Tue   -109.  spring
#>  2 2013-01-02   943 Wed    -19.7 spring
#>  3 2013-01-03   914 Thu    -51.8 spring
#>  4 2013-01-04   915 Fri    -52.5 spring
#>  5 2013-01-05   720 Sat    -24.6 spring
#>  6 2013-01-06   832 Sun    -59.5 spring
#>  7 2013-01-07   933 Mon    -41.8 spring
#>  8 2013-01-08   899 Tue    -52.4 spring
#>  9 2013-01-09   902 Wed    -60.7 spring
#> 10 2013-01-10   932 Thu    -33.8 spring
#> # ... with 355 more rows
```

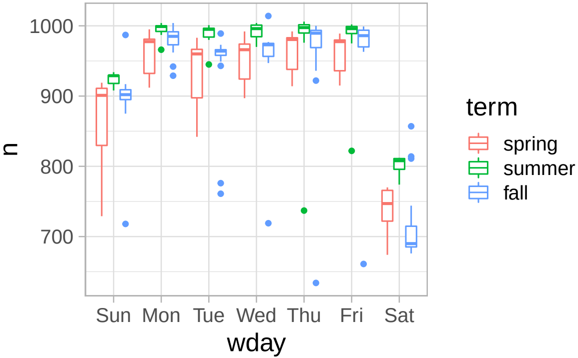
The three school terms cont'd


```r
daily %>%
  filter(wday == "Sat") %>%
  ggplot(aes(date, n, color = 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

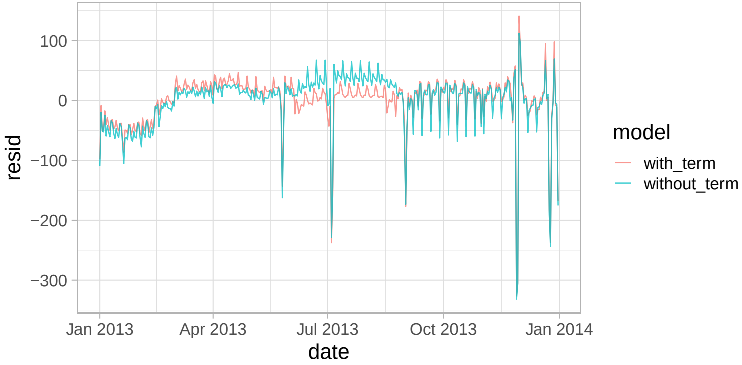
```r
daily %>%
  ggplot(aes(wday, n, color = term)) +
  geom_boxplot()
```



An improved model

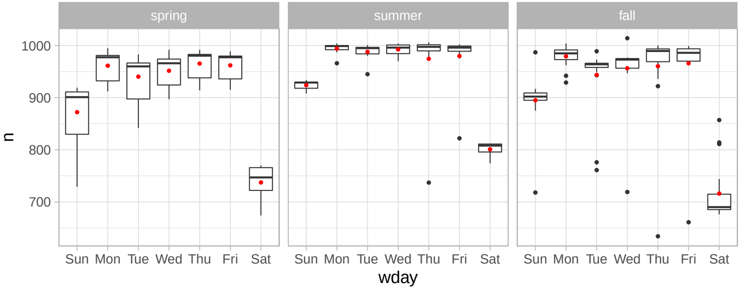
Columbia University in the City of New York

```
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, color = model)) +
  geom_line(alpha = 0.75)
```



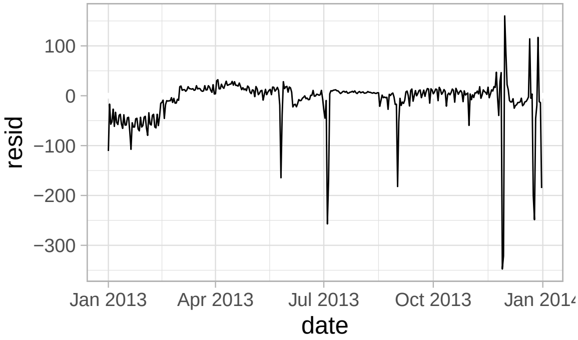
What's going on here?

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

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



Robust fit

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

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



Computed variables

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- 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

Columbia University in the City of New York

```r
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, color = wday)) +
  geom_line() +
  geom_point()
```