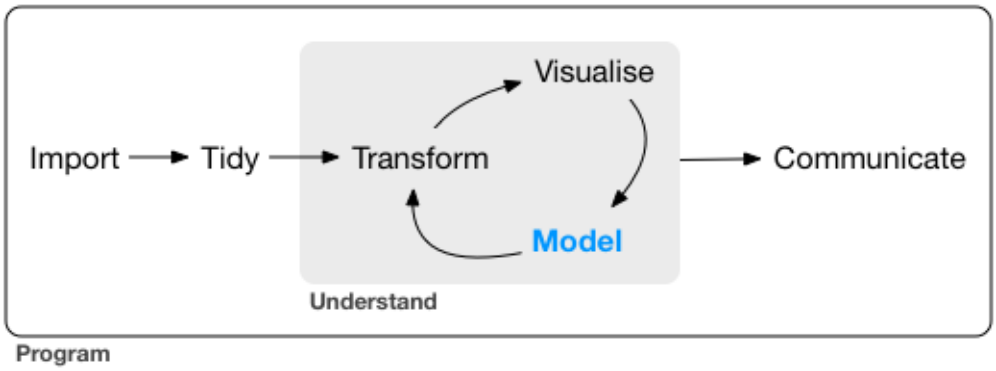
Wicham On Predictive Statistical Models

Big picture paradigm

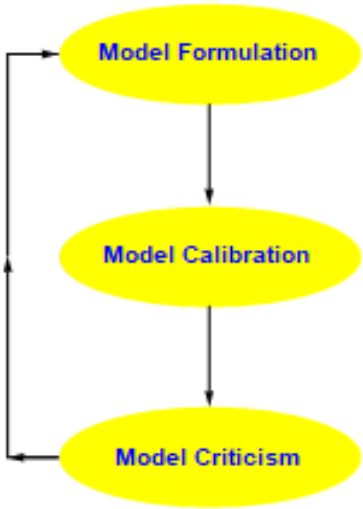


The trifecta of the tools for EDA.

How to visualise more complex model types, you might try <http://vita.had.co.nz/papers/model-vis.html> LINK.

Classical paradigm

Box and Jenkins (1970, 2015). R.A. Fisher (1925)



Wicham paradigm

Variations are widely used in Machine Learning starting with training neural nets in the 1990's.

- 60% data ⟹ training set
- 20% data ⟹ query (or validation) set
- 20% data ⟹ test set

1990's neural net and most Machine Learning

- 1/3 training, 1/3 validation, 1/3 test

Simple approach

- 2/3 data ⟹ training set
- 1/3 data ⟹ test set

G. E. P. Box Quotes

1. All models are wrong, but some are useful.
2. Now it would be very remarkable if any system existing in the real world could be exactly represented by any simple model. However, cunningly chosen parsimonious models often do provide remarkably useful approximations. For example, the law PV = RT relating pressure P, volume V and temperature T of an "ideal" gas via a constant R is not exactly true for any real gas, but it frequently provides a useful approximation and furthermore its structure is informative since it springs from a physical view of the behavior of gas molecules.
   For such a model there is no need to ask the question "Is the model true?". If "truth" is to be the "whole truth" the answer must be "No". The only question of interest is "Is the model illuminating and useful?".
3. Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.

Exploring a simple models

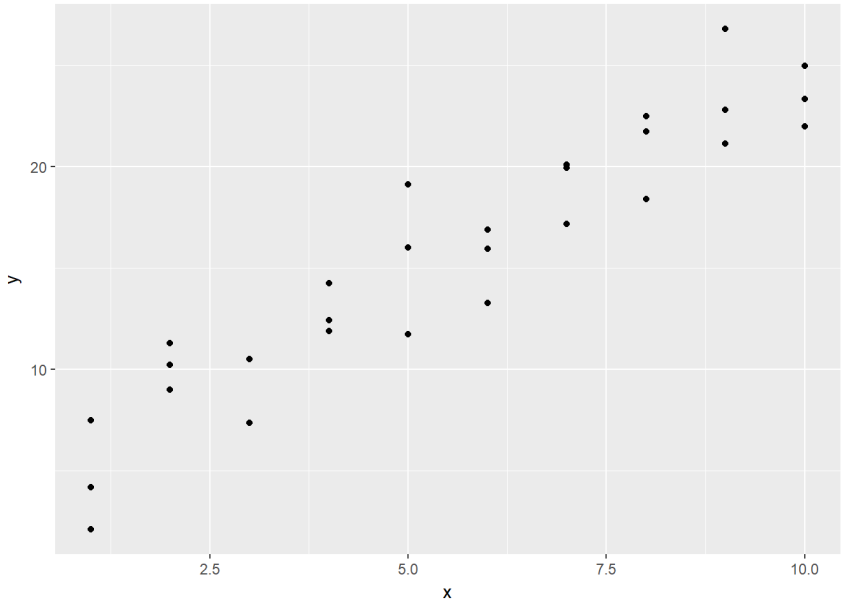
Discussion is given for fitting simple least-squares regression models but the principle discussed for much more general setups such as GLM, robust regression and many other parametric models.

sim1, sim2, sim3 and sim4 are simple simulated examples included in the modelr package.

```
> sim1
# A tibble: 30 x 2
       x         y
```

```
   <int>     <dbl>
 1     1  4.199913
 2     1  7.510634
 3     1  2.125473
 4     2  8.988857
 5     2 10.243105
 6     2 11.296823
 7     3  7.356365
 8     3 10.505349
 9     3 10.511601
10     4 12.434589
# ... with 20 more rows
```

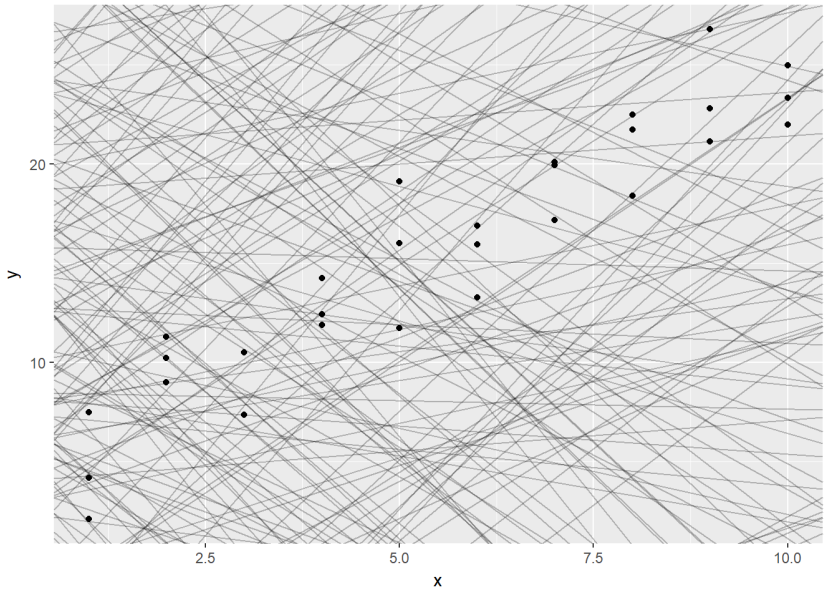
```
ggplot(sim1, aes(x, y)) +
  geom_point()
```



First we consider simple linear regression models, $y = a_1 + a_2\, x + \epsilon$.

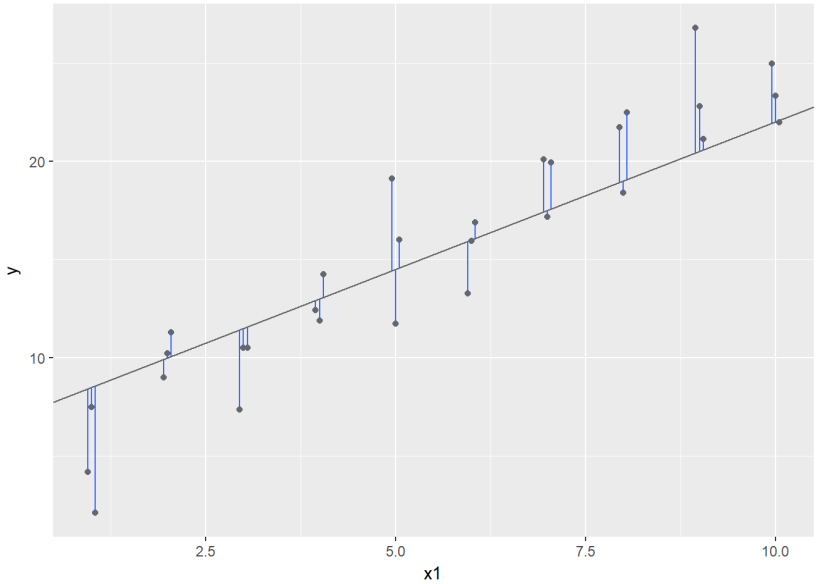
Let's consider many random candidate models.

```
models <- tibble(
  a1 = runif(250, -20, 40),
  a2 = runif(250, -5, 5)
)

ggplot(sim1, aes(x, y)) +
  geom_abline(aes(intercept = a1, slope = a2), data = models, alpha = 1/4) +
  geom_point()
```




```
dist1 <- sim1 %>%
  mutate(
    dodge = rep(c(-1, 0, 1) / 20, 10),
    x1 = x + dodge,
    pred = 7 + x1 * 1.5
  )

ggplot(dist1, aes(x1, y)) +
  geom_abline(intercept = 7, slope = 1.5, colour = "grey40") +
  geom_point(colour = "grey40") +
  geom_linerange(aes(ymin = y, ymax = pred), colour = "#3366FF")
```

The vertical distance from the point to the line = RESIDUAL



The LS criterion defines the parameters by the equation,

$$(\hat{a}_1, \hat{a}_2) = \underset{\dot{a}_1,\dot{a}_2}{\text{argmin}} \sum_{i=1}^{n} e_i^2 = \underset{\dot{a}_1,\dot{a}_2}{\text{argmin}} \sum_{i=1}^{n} (y_i - \dot{a}_1 - \dot{a}_2 x_i)^2 \quad (1)$$

This criterion is optimal in the sense of maximum likelihood when the errors are normally distributed. But in many cases, especially with large datasets containing outliers a better criterion is least trimmed squares (LTS, Wikipedia),

$$(\hat{a}_1, \hat{a}_2) = \underset{\dot{a}_1,\dot{a}_2}{\text{argmin}} \sum_{i=1}^{k} e_{(i)}^2 \quad (2)$$

where $k = [n/2]$, $e_{(1)}^2 \leq e_{(2)}^2 \leq \ldots \leq e_{(n)}^2$ is the order statistic and k is selected less than n. The LTS is highly robust to wild outliers. It is said to be resistant because even when some data are allowed to increase in absolute value indefinitely the estimates continue to track the true model. The LTS optimization problem in (2) is computationally very difficult so more complex robust regression methods that are both computationally efficient and highly robust and resistance to wild values are been developed. The state of the art robust regression estimates are available in the R recommended package robustregression.

An approximate solution is easily obtained for either LS or LTS by simple enumeration of the tentative models shown in the Figure above. For LS, exact solution is

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

modr functions

- **modelr::data_grid()** - To visualise a model, it is very useful to be able to generate an evenly spaced grid of points from the data. data_grid helps you do this

```
> data_grid(mtcars, vs, am)
# A tibble: 4 x 2
     vs    am
  <dbl> <dbl>
1     0     0
2     0     1
3     1     0
4     1     1
> mtcars$vs
 [1] 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1
> mtcars$am
 [1] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1
```

base::expand.grid(): Create a data frame from all combinations of the supplied vectors or factors.

```
> expand.grid(0:1, 10:11)
  Var1 Var2
1    0   10
2    1   10
3    0   11
4    1   11
```

- **modelr::add_predictions()** / **modelr::add_residuals()** takes a data frame and a model. It adds the predictions/residuals from the model to a new column in the data frame
- **modelr::model_matrix()** simply an alternative to **base::model.matrix()**

Reference: "Symbolic Description of Factorial Models for Analysis of Variance" by G. N. Wilkinson and C. E. Rogers <https://www.jstor.org/stable/2346786>

More advanced models

- Generalised linear models: `stats::glm()`. Linear models assume that the response is continuous and the error has a normal distribution. Generalised linear models extend linear models to include non-continuous responses (e.g. binary data or counts). They work by defining a distance metric based on the statistical idea of likelihood.
- Generalised additive models: `mgcv::gam()`, extend generalised linear models to incorporate arbitrary smooth functions. That means you can write a formula like `y ~ s(x)` which becomes an equation like `y = f(x)` and let `gam()` estimate what that function is (subject to some smoothness constraints to make the problem tractable).
- Penalised linear models__, e.g. `glmnet::glmnet()`, add a penalty term to distance that penalises complex models (as defined by the distance between the parameter vector and the origin). This tends to make models that generalise better to new datasets from the same population.
- Robust linear models__, e.g. `MASS:rlm()`, tweak the distance to downweight points that are very far away. This makes them less sensitive to the presence of outliers, at the cost of being not quite as good when there are no outliers. I recommend the robustbase package.
- Trees, `rpart::rpart()` and C50 attack the problem in a completely different way than linear models. They fit a piece-wise constant model, splitting the data into progressively smaller and smaller pieces. Trees aren't terribly effective by themselves, but they are very powerful when used in aggregate by models like random forests, `randomForest::randomForest()` or gradient boosting machines `xgboost::xgboost`.