Statistical Rethinking: Lecture 03

with Python code

Linear Regression: statistics' geocentric models

Geocentric models, such as Ptolomy's, could account very well to explain the motion of planets, even though they're plainly wrong. Planets do not move along epicycles, yet, this device is useful for prediction. The same is true for Linear Regression models. Variables and their generative processes are rarely linear, error distributions are rarely Gaussian. Yet, the model is very useful if handed with care. It describes associations, allows to make predictions, but it is mechanistically wrong.

Interestingly, the history of Linear Regression is intimately related to astronomy. Gauss developed least-squares linear regression and introduced the Gaussian curve for predicting the motion of a, at the time, comet. Now we know it was the dwarf planet Ceres. It worked. His methods were later rediscovered in the 20th century.

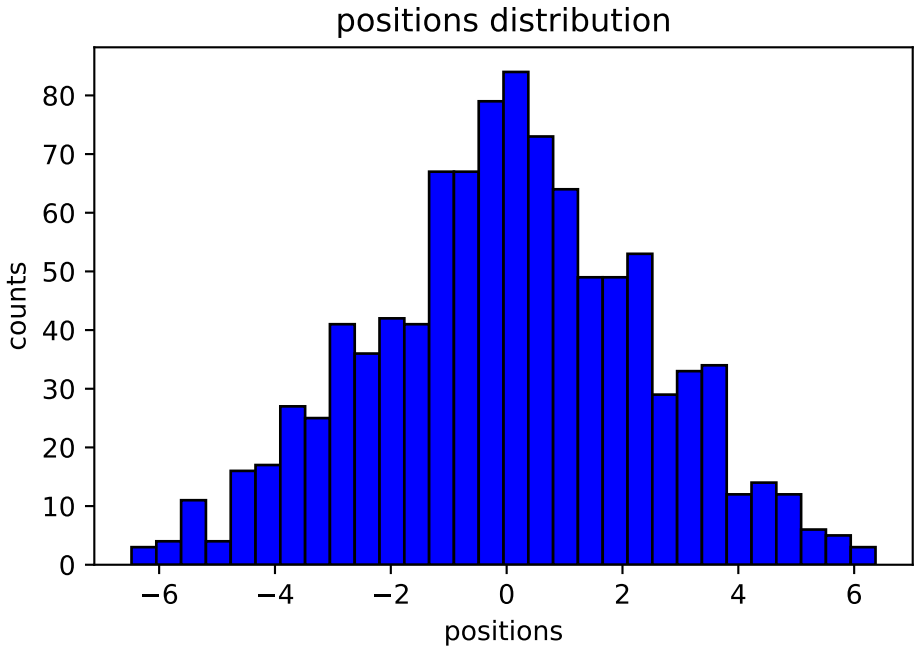
Normal Distributions

The easyiest ways to generate a Gaussian distribution is by summing up random variables. As the number of random trials grow, the sum is approaches a normal distribution. This is the central limit theorem.

Assume you and a 1000 of your closest friends are all aligned along a line in the floor. Each one of you flip a coin. If its tails you move to the right. If it's heads you move to the left. If we plot the distribution of distances from the line everyone started from, it would look like this

```
# generate the coin tosses (uniform), and sum up
pos = np.sum(np.random.uniform(-1, 1, (1000, 16)), axis=1)
```

```
# plot results
mplt.hist(pos, bins=30, color='blue', edgecolor='black')
mplt.xlabel('positions')
mplt.ylabel('counts')
mplt.title('positions distribution')
mplt.show()
```



Increase the number of tosses.

Why do we use Normal distributions? Because most of the times we can justify their use by the variables generative processes: accumulation of fluctiations. Also, there is an inferential argument. If we are interested in estimating mean and variance of a variable, the normal distribution is the least informative, most parsimonious choice, according to the theoretic-information principle of maximum entropy. Most importantly, the varibales have not to be normally distribution for a normal model to be useful. And Gaussians are very useful because we can do a lot o otherwise complicated calculations (conditioning, marginalizing) much more simpler.

The goals of this lecture are 1) to introduce language for representing models, 2) calculate posterior distributions with multiple unknowns, 3) constructing and understanding linear models.

Workflow

1) state a clear question (estimand)
2) sketch causal assumptions (scientific model)
3) use the sketch to define a generative model (statistical model)
4) use the generative model to build estimator (validate model)
5) apply to real samples

Gaussian Distribution

Recall that stating $y \sim \text{Normal}(\mu, \sigma)$ means

$$\Pr(y|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y-\mu)^2}{2\sigma^2})$$

and it is a contiuous density function, in contrast with the Binomial distribution from the previous lecture, which already a probability mass function. Masses are obtanined from the area below densities.

The influence of height on weigth

We want answer how does height influence weight? For this we will analyze datasets containing these variables. The scientific prior knoledge for the variables generative process is that changes in height result in changes in weight. Gaussian variations in height result from the growth history: accumulated fluctuations in growth.

The causal model for the variables is the following: weight, W, is influenced both by height H and by the overall effect of unobserved variables, U. We can write

$$W = \beta H + U.$$

Coding it up in a function

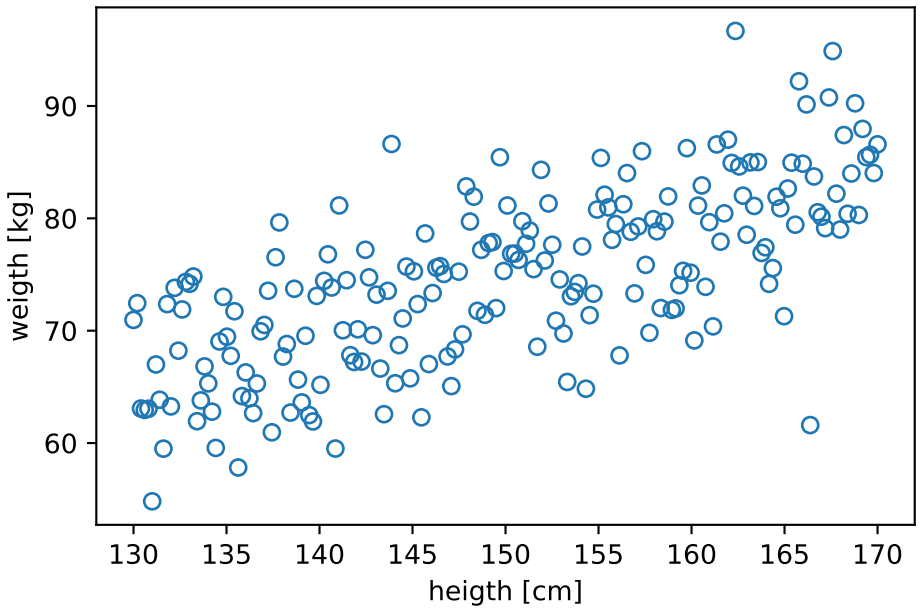
```
def simulate_weigth(height, beta, std):
    U = np.random.normal(loc=0, scale=std, size=height.size)
    return beta * H + U
```

U effectively acts as a background noise. Next we generate some 200 synthetic people with heigths ranging from 130 to 170 centimeters, and compute their weigth

```
H = np.linspace(130, 170, 200)
W = simulate_weigth(H, beta=0.5, std=5)
```

plotting

```
mplt.figure()
mplt.plot(H, W, 'o', mfc='none')
mplt.xlabel('heigth [cm]')
mplt.ylabel('weigth [kg]')
mplt.show()
```



Describing models

The above modeling can be described as

$$W_i = \beta H_i + U_i,$$

$$U_i \sim \text{Normal}(0, \sigma)$$

$$H_i \sim \text{Uniform}(130, 170)$$

The `simulate_weigth` function is our generative model encapsulating the scientific model. Now we focus on the statistical model, or estimator.

The estimator

The estimator will be the expected Weight at a given height, for which we assume a linear model

$$\mathbb{E}[W_i|H_i] = \alpha + \beta H_i$$

the posterior will be given by

$$\Pr(\alpha, \beta, \sigma|W_i, H_i) = \frac{1}{Z}\Pr(W_i|H_i, \alpha, \beta, \sigma)\Pr(\alpha, \beta, \sigma)$$

since the posterior is over the parameters of the regression model, it actually consists on the distribution for the lines describing the linear associations.

this modeling leads to

$$W_i \sim \text{Normal}(\alpha + \beta H_i, \sigma)$$

TODO: generate observations with $\sigma = 10$, $\beta = 0.5$. Compute the posterior for a grid of 11 β values. Plot the posterior. Project the distribution in outcome space (plot the resulting lines)

Next, generalize and estimate α as well. Plot the distribution density in the parameter α β space.