Lab2

PSTAT 115, Winter 2022

Jan 19, 2023

Goal:

- Brief review: sufficient statistics.
- An example of data generating process (DGP).
- An example of Bayesian analysis.

Review

Sufficient Statistics:

- Let $L(\theta) = p(y_1, y_2, ..., y_n|\theta)$ be the likelihood and $s(y_1, ..., y_n)$ be a statistic
- factorization theorem: s is sufficient if $L(\theta) = h(y_1, ..., y_n)g(s, \theta)$, where h is not a function of θ
- $L(\theta) \propto g(s, \theta)$
- Many possible sufficient statistics
  - Often seek a minimal sufficient statistic

- Let $X_1, X_2, \ldots, X_n$ $i.i.d \sim N(\mu, \sigma^2)$ with known variance.

$$\begin{aligned}
L(\mu, \sigma) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \\
&= \left(2\pi\sigma^2\right)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right) \\
&= \left(2\pi\sigma^2\right)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}x_i^2 - 2\mu\sum_{i=1}^{n}x_i + n\mu^2\right)\right) \\
&= \left(2\pi\sigma^2\right)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}x_i^2\right) \exp\left(-\frac{1}{2\sigma^2}\left(-2\mu\sum_{i=1}^{n}x_i + n\mu^2\right)\right) \\
&= h(x_1, ..., x_n)g(T, (\mu, \sigma^2))
\end{aligned}$$

Since σ^2 is known. i.e.

$$g(T, (\mu, \sigma^2)) = \exp\left(-\frac{1}{2\sigma^2}\left(-2\mu\sum_{i=1}^{n}x_i + n\mu^2\right)\right)$$

where

$$T = \sum_{i=1}^{n} x_i$$

and

$$h(x_1, ..., x_n) = \left(2\pi\sigma^2\right)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} x_i^2\right)$$

And $\sum_{i=1}^{n} X_i$ is a sufficient statistic.

Data Generating Process (DGP)

Dataset and Background

If you ever want to just experiment with numbers, there are a bunch of handy datasets built into common R packages. To see them type:

```
data()
```

Let's use one of these to talk about generating models + covariates.

ChickWeight is a dataset of weights collected over time for a number of different chicks on different diets. Let's load up the ChickWeight dataset into a tibble to make it easy to work with:

```
df = ChickWeight %>% as_tibble
```

The %>% notation is a pipe in the tidyverse package. It is shorthand for:

```
df = as_tibble(ChickWeight)
```

Using tibbles (instead of base R dataframes) and pipes (instead of function calls) can make it easier to do basic data transformations. You can do all of this in base R as well.

First let's print the ChickWeight tibble and see what's in it:

```
df
```

```
## # A tibble: 578 x 4
##    weight  Time Chick Diet
##     <dbl> <dbl> <ord> <fct>
##  1     42     0 1     1
##  2     51     2 1     1
##  3     59     4 1     1
##  4     64     6 1     1
##  5     76     8 1     1
##  6     93    10 1     1
##  7    106    12 1     1
##  8    125    14 1     1
##  9    149    16 1     1
## 10    171    18 1     1
## # ... with 568 more rows
```

Usually we'd look at the data before writing out a generating model. Just as an exercise though, let's do the reverse. We'll write out a generating model and then make some plots to see how reasonable our assumptions were.

- What is it that we're trying to model?
  - Weight
- What other data did we collect to explain what we're modeling (covariates)?
  - Time

    - Chicken ID
    - Diet

The simplest thing we might assume is that errors in our measurements are normally distributed. This means:

$$\text{weight} \sim N(\mu, \sigma)$$

In this case, weight is fixed data, and we'll need to estimate μ and σ (they are the estimands).

- What might be a limitation of the normal assumption here?
    - Our chickens grow over time. Lumping all the weights together at these different times won't produce a normal distribution.
    - There are multiple diets. Should every diet have the same mean?
    - Our outcomes appear to be rounded (looks like we'd never measure a weight of 75.33333 for instance)
    - Weights will be non-negative

How might we incorporate the chickens growing over time? We might assume that chickens grow linearly with time:

$$\text{weight}_i \sim N(\alpha t_i + \beta, \sigma)$$

In this case the new estimands are α, β, and σ. weight_i and t_i are both data.

How might we expect the different diets to affect things? Maybe it doesn't affect the time zero weight of the chicken (β), but it does affect the slope (α)? We can code in a diet dependence easily enough:

$$\text{weight}_{i,d} \sim N(\alpha_d t_i + \beta, \sigma)$$

The generating model also requires us to specify how the chickens are given a diet. We might assume this is uniformly sampled from the available diets, or something else. This is more important if we actually need to estimate the diet the chicken ate. In this case, the diet is given.

Pseudocode

Assume that the diet is categorically distributed with 4 categories and corresponding probabilities $p = (p_1, p_2, p_3, p_4) = (0.4, 0.2, 0.2, 0.2)$. The covariate *time* and parameters $\alpha_1, \cdots, \alpha_k$, β are given.

```
for(i in 1:nrow(ChickWeight)) {
  Draw diet_i from Categorical(4, p)
  Set alpha_i = alpha_k if diet_i = k, k = 1, 2, 3, 4
  Draw weight_i from Normal(alpha_i * t_i + beta, sigma)
}
```

```
library(pracma)
```

```
##
## Attaching package: 'pracma'

## The following object is masked from 'package:purrr':
##
##     cross
```

```
p=c(0.4, 0.2, 0.2, 0.2)
alpha_param=c(8, 10, 12, 15)
beta_param=3

diet_ex = randsample(4,nrow(ChickWeight),w=p,replacement = TRUE)
```

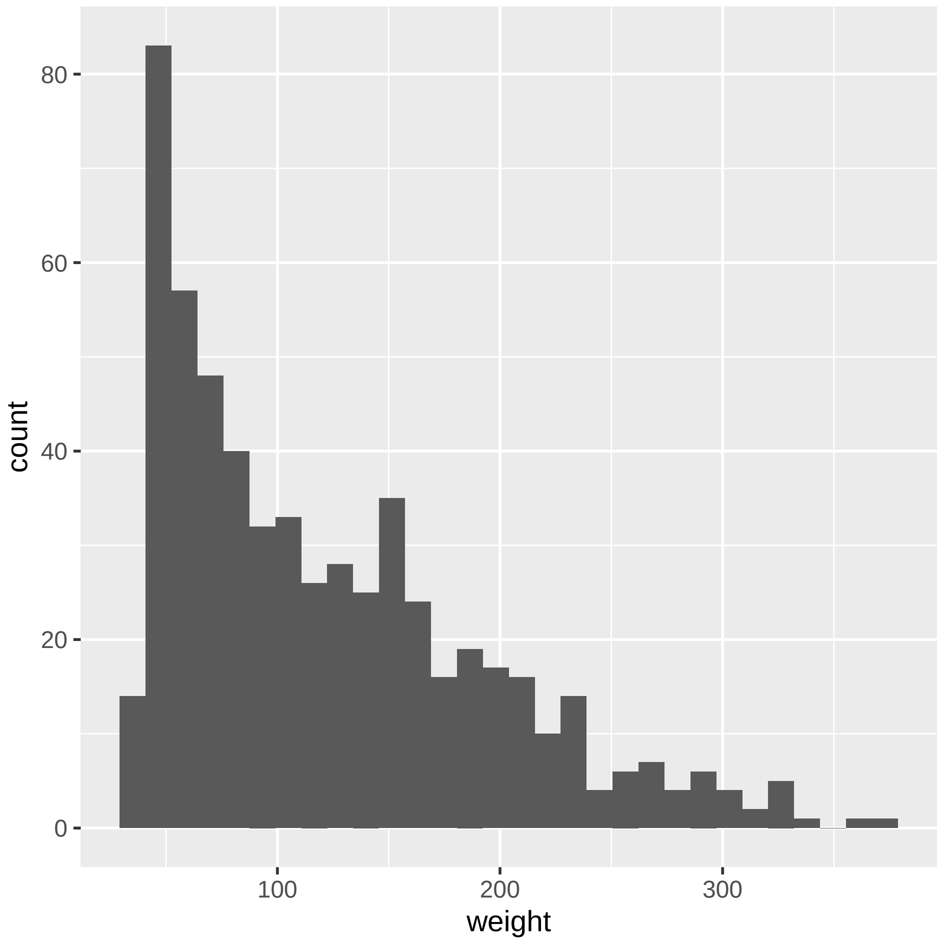
```
alpha_ex = alpha_param[diet_ex]

weight_ex = numeric()
for (i in 1:nrow(ChickWeight)){
  weight_ex[i] = rnorm(1,alpha_ex[i]*df$Time[i]+beta_param, 1)
}
```

Exploring our data

Since it is the weights we are concerned with, let's look at that first.

```
df %>%
  ggplot(aes(weight)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

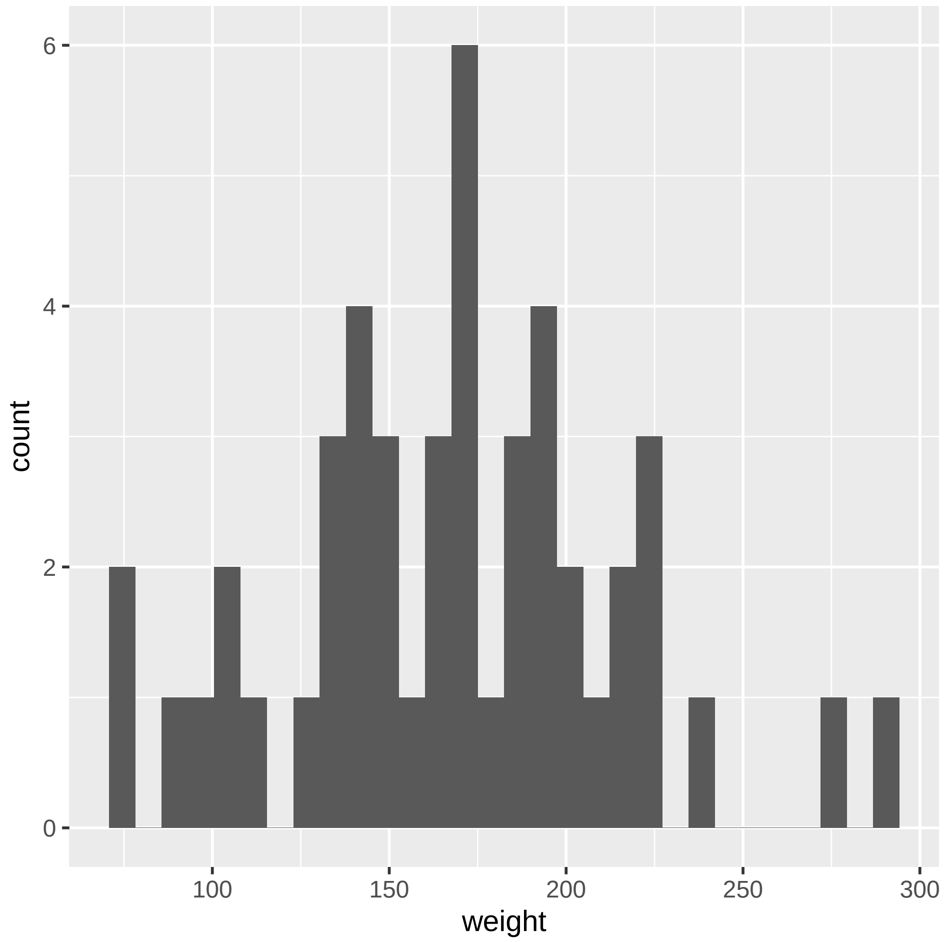


Perhaps unsurprisingly, this plot is mostly unintelligible. There's all sorts of things mixed together, so let's try to use some more plots to break it apart.

Let's have a look at $p(\text{weight}|\text{Time} = t)$.

```
df %>%
  filter(Time == 16) %>%
  ggplot(aes(weight)) +
  geom_histogram()
```

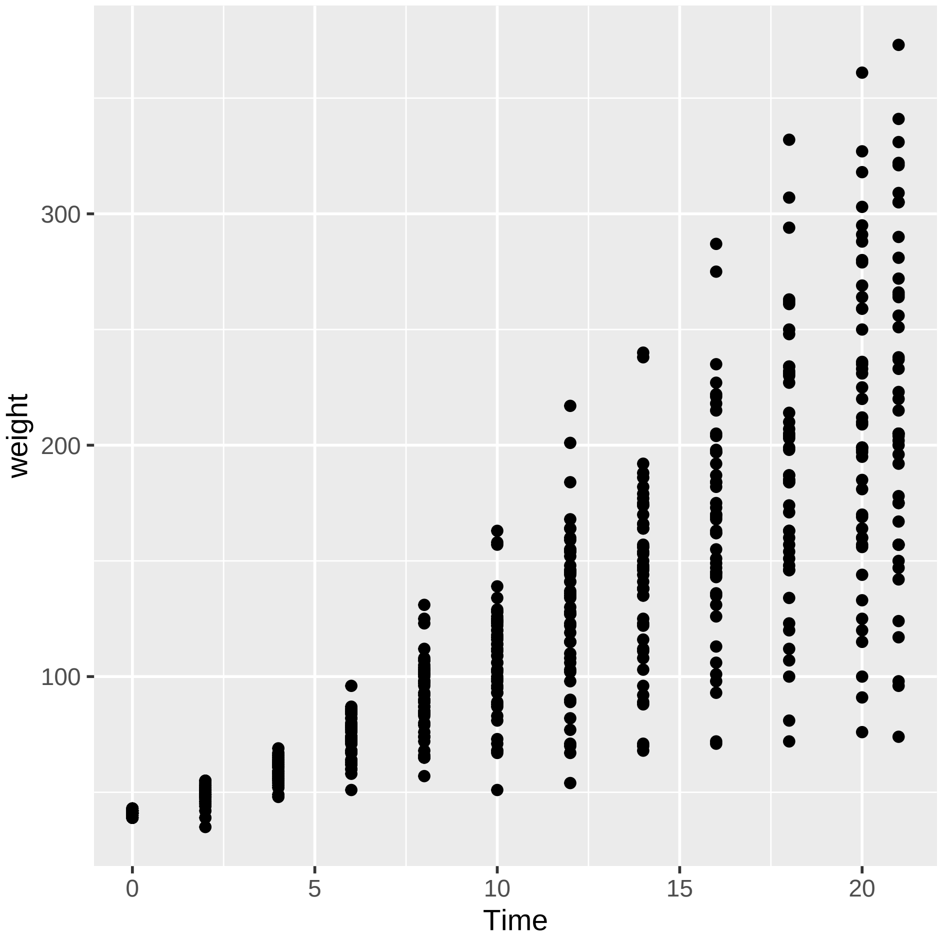
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



If we assume this conditional distribution is normal, then we're saying the output (weight), is characterized by two parameters μ and σ (that can be functions of the covariates). μ and σ are our estimands here – the things we'd try to estimate.

Let's try to get an idea of what $\mu(t)$ would look like by plotting weight vs. time:

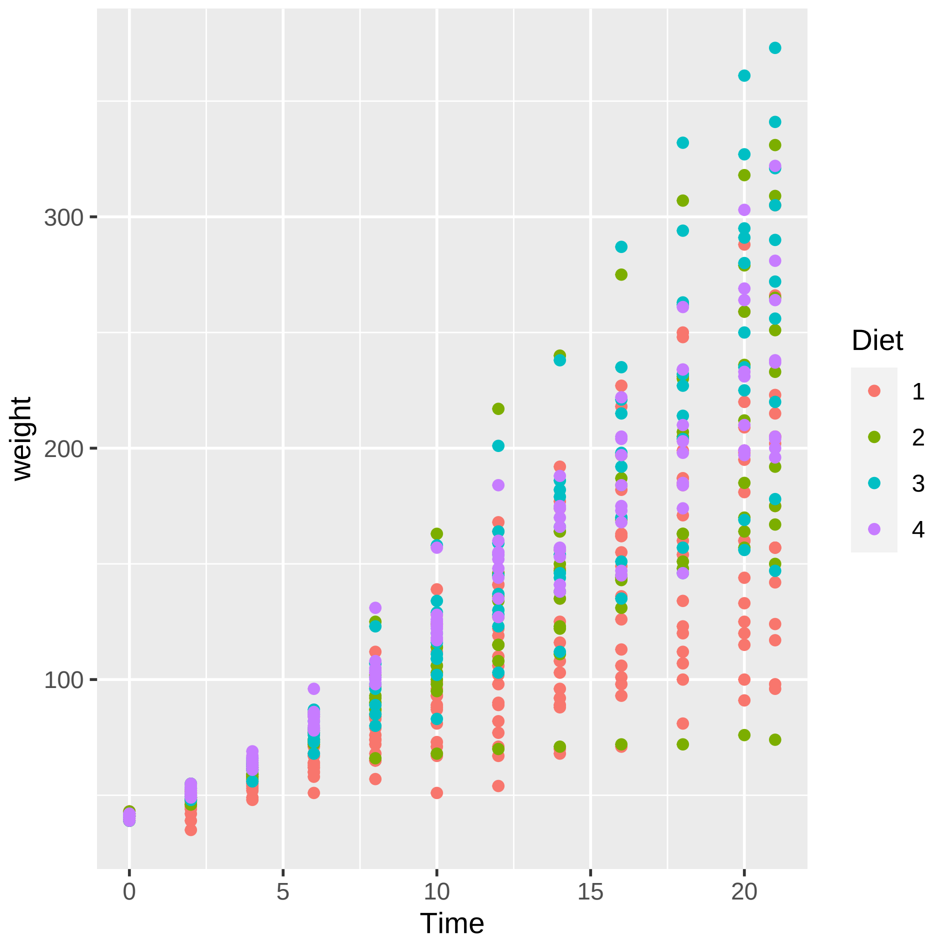
```
df %>%
  ggplot(aes(Time, weight)) +
  geom_point()
```



- How does this differ from what we proposed initially?
  - The noise is heteroskedastic (this means the amount of noise changes with another covariate – in this case time)

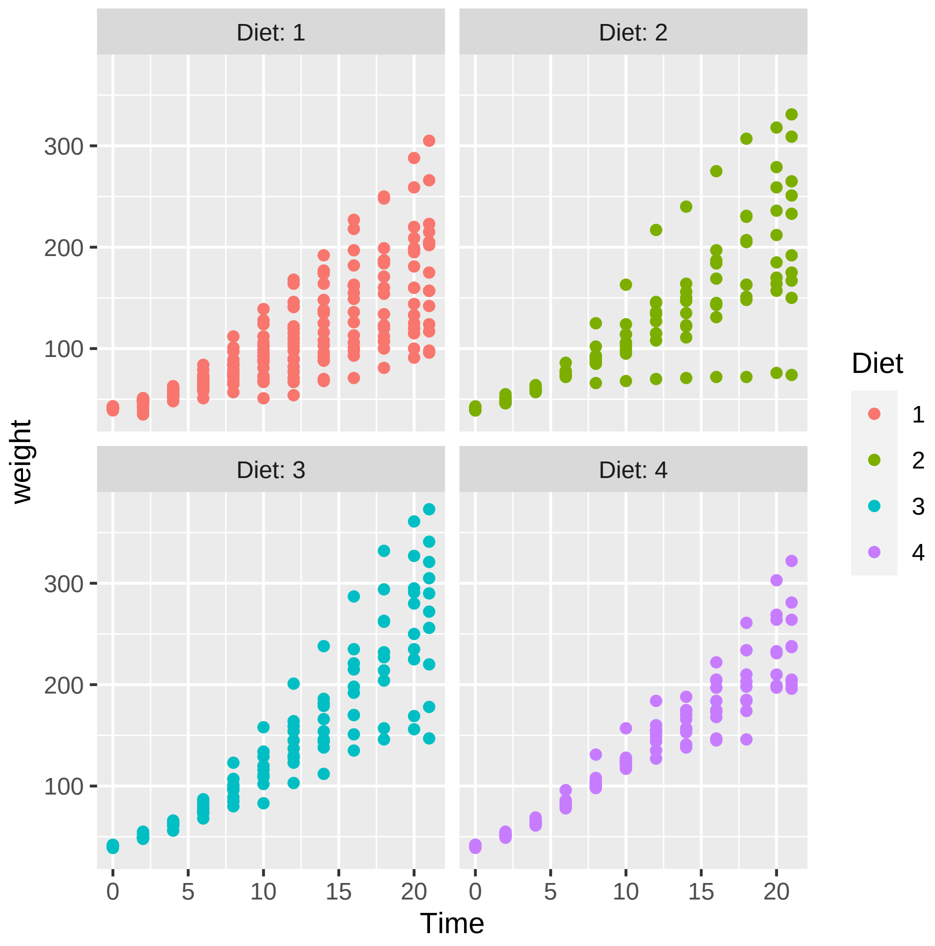
Let's try to get a handle on how diet affects any of this:

```
df %>%
  ggplot(aes(Time, weight)) +
  geom_point(aes(color = Diet))
```



That's kindof hard to see. Maybe we can break this out into different plots:

```
df %>%
  ggplot(aes(Time, weight)) +
  geom_point(aes(color = Diet)) +
  facet_wrap(~ Diet, nrow = 2, labeller = label_both)
```



- Does diet seem to have a strong affect on the growth of the chickens?
  - The effect isn't clear to me. Maybe someone else is more creative. At best maybe diet 4 produces more consistently sized chickens?
- What might be the next step in this analysis?
  - I'd feed food to more chickens. It's probably not that expensive to collect more data here.

Bayesian Example

An example from BDA3 (the Gelman book in the syllabus) is a spelling correction problem. The problem setting is, assume that a user searched for "radom". The question is, what is the probability that they typed something else?

Look at lecture slides defining posterior.

First assume that there are only three words to consider, "radom", "random", and "radon". Assume also that you've been given the likelihoods for typing "radom" given you actually wanted to type each of those three words and also the prior probability that each of those words was what was typed.

Likelihoods:

$$p(\text{typed radom}|\text{wanted to type random}) = 0.00193$$

$$p(\text{typed radom}|\text{wanted to type radon}) = 0.000143$$

$$p(\text{typed radom}|\text{wanted to type radom}) = 0.975$$

Priors:

$$p(\text{wanted to type random}) = 7.60 * 10^{-5}$$
$$p(\text{wanted to type radon}) = 6.05 * 10^{-6}$$
$$p(\text{wanted to type radom}) = 3.12 * 10^{-7}$$

Now we can use Bayes' rule to compute the posterior.

```r
likelihood = c(0.00193, 0.000143, 0.975)
prior = c(7.60e-5, 6.05e-6, 3.12e-7)

unnormalized_posterior = likelihood * prior
posterior = unnormalized_posterior / sum(unnormalized_posterior) #turns into percentages
posterior
```

```
## [1] 0.324696347 0.001915128 0.673388524
```

Posterior:

$$p(\text{wanted to type random}|\text{typed radom}) = 0.325$$
$$p(\text{wanted to type radon}|\text{typed radom}) = 0.002$$
$$p(\text{wanted to type radom}|\text{typed radom}) = 0.673$$

Questions

- Was this what you expected?
  - The probability of no error seems high. Radom is a city in Poland though. Maybe it is a more popular topic of discussion than we might expect?
- How do you think the prior was estimated?
  - Empirical counts
- How do you think the likelihood was estimated?
  - Perhaps there was a study for this? Maybe a researcher sat hundreds of people down on keyboards and tried to understand the types of spelling mistakes that were more or less common