Thoughts on Visualisations And the Joy of Grammars

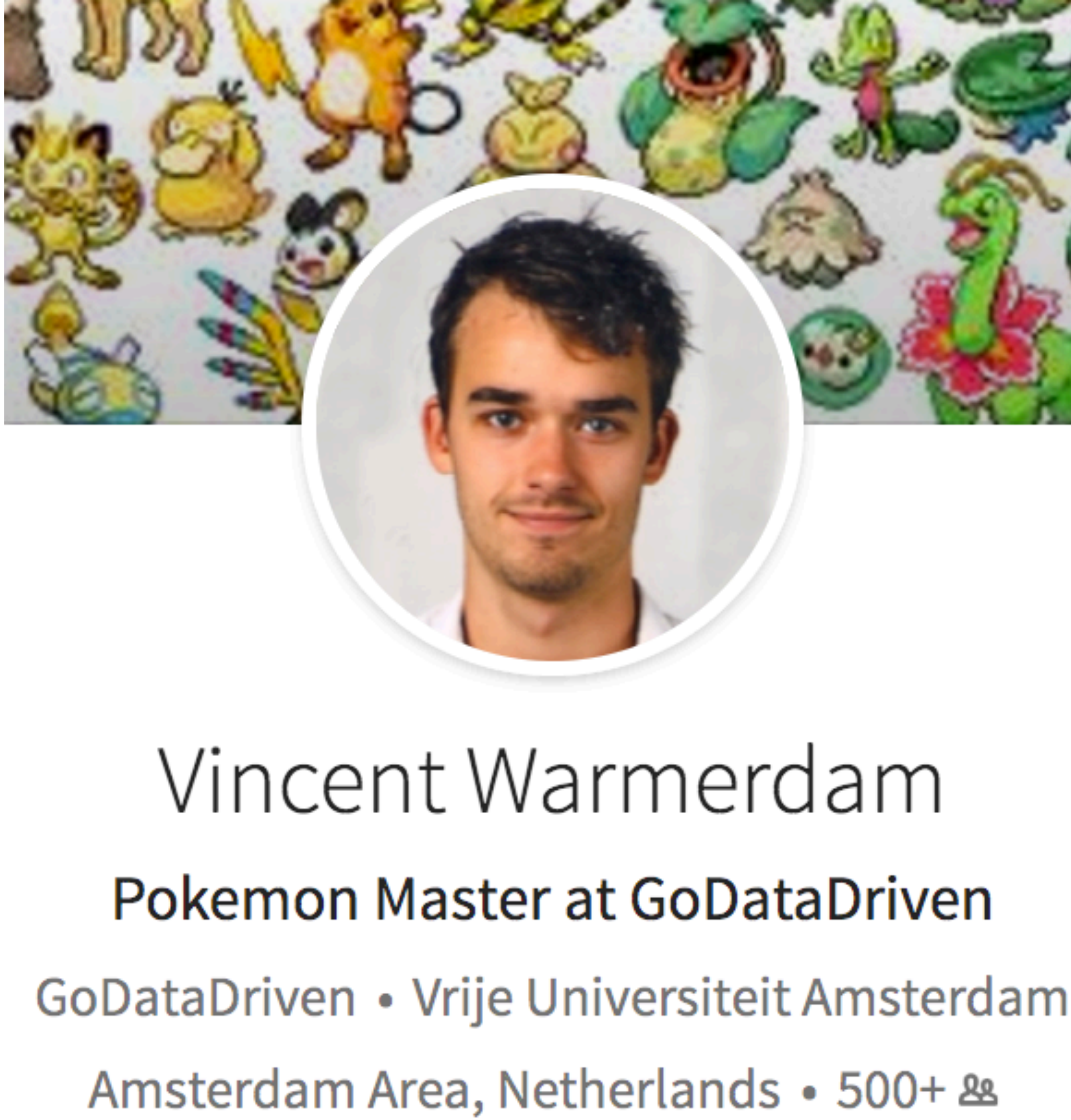
Vincent D. Warmerdam

koaning.io - fishnets88 - GoDataDriven

Who is this guy?

3 years @ GDD
koaning.io
PyData Amsterdam
Rstudio partner
Machine Learning Meetup
free open sessions in data
18 endorsements for awesomeness
founder @ tnaas.com

I write code,
I solve data problems,
ask me anything.



Vincent Warmerdam

Pokemon Master at GoDataDriven

GoDataDriven • Vrije Universiteit Amsterdam

Amsterdam Area, Netherlands • 500+

TL;DR

Keep it simple and keep it boring whenever possible. Be careful that you don't turn data viz into info porn.

Try to seperate concerns; data crunching and visualisation might be two seperate steps.

Optimise time to payback, omit complexity. Be very careful with interactivity. End user may be n00b.

MLaaUI may become a thing.

Today

- I'll demonstrate why we like data viz
  - monopoly example
  - lego example
  - radial basis function
- I'll explain a nice sniff test for viz
  - information vs ink ratio
- I'll demonstrate why we like grammars
  - grammar for data manipulation
  - grammar for data visualisation
- I'll conclude with examples where interactivity helps
  - biased mercartor maps
  - inverse turing test
  - machine learning as a user interface

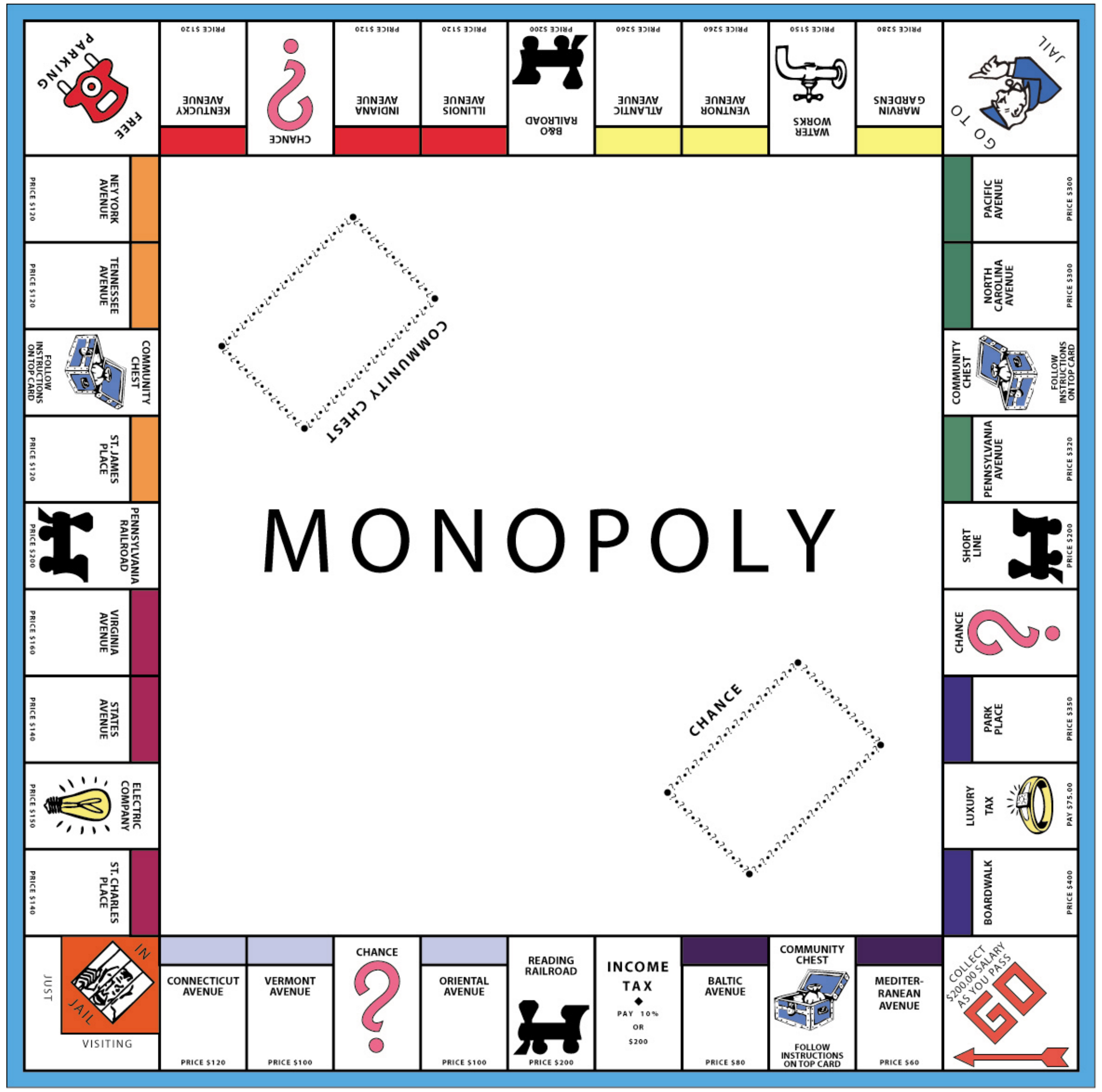
Let's talk data viz

The only professional reason why I am interested in visualisation is that it helps keep communication clear and simple.

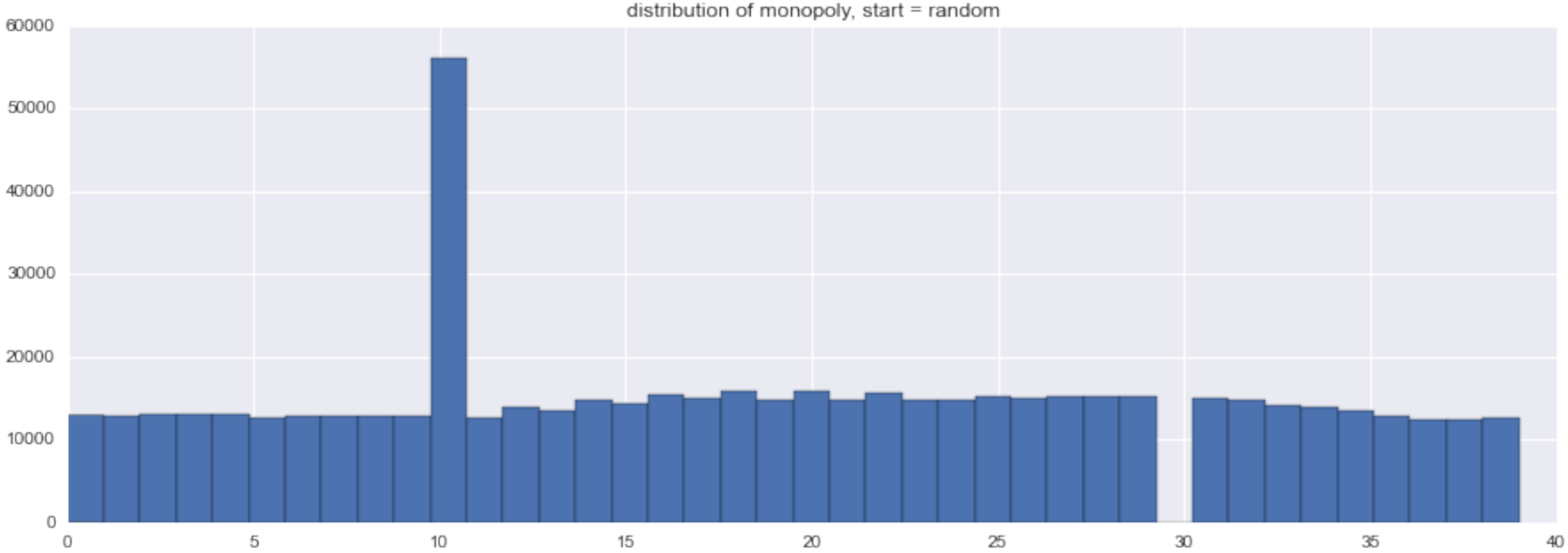
If a data viz is not clear or not simple, I cannot use it.

I will give two examples of slightly complex analyses that are much easier to explain with visualisation.

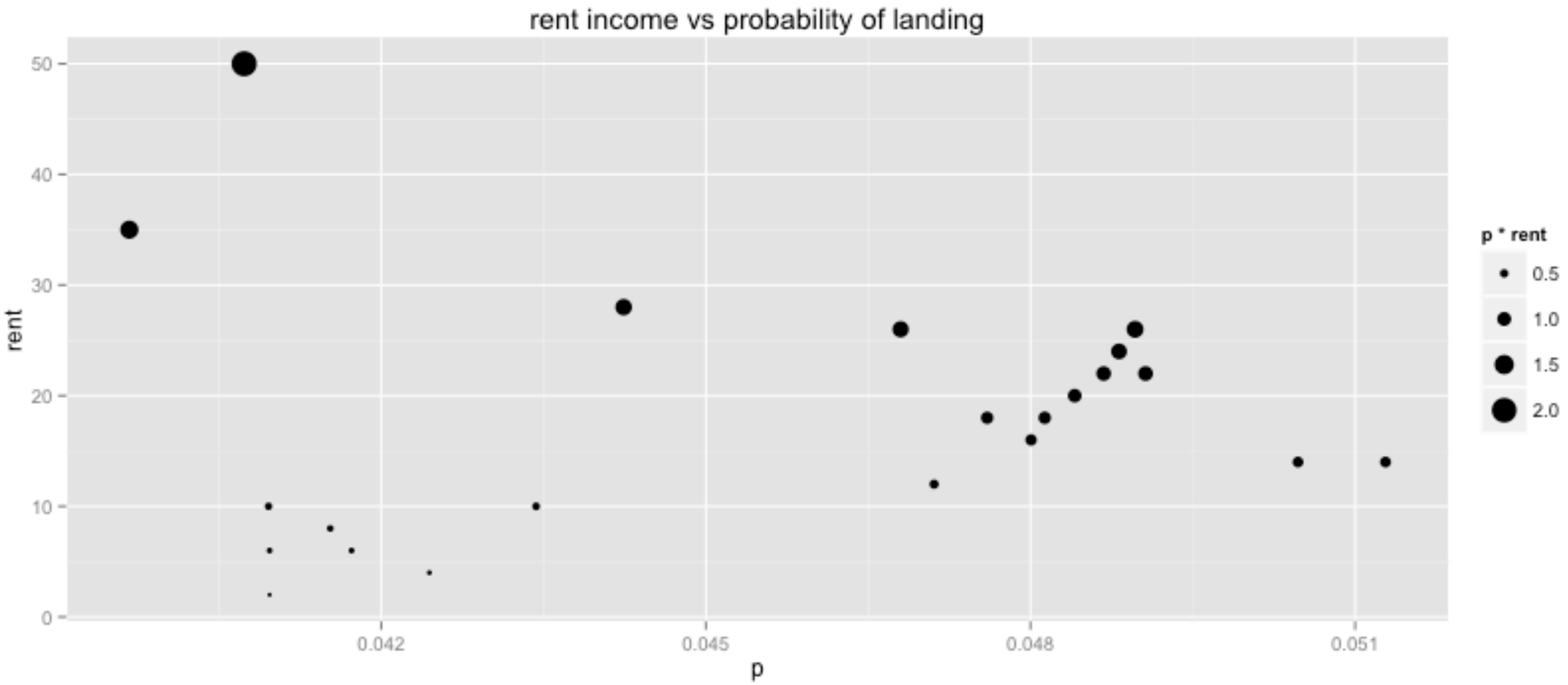
Monopoly Example



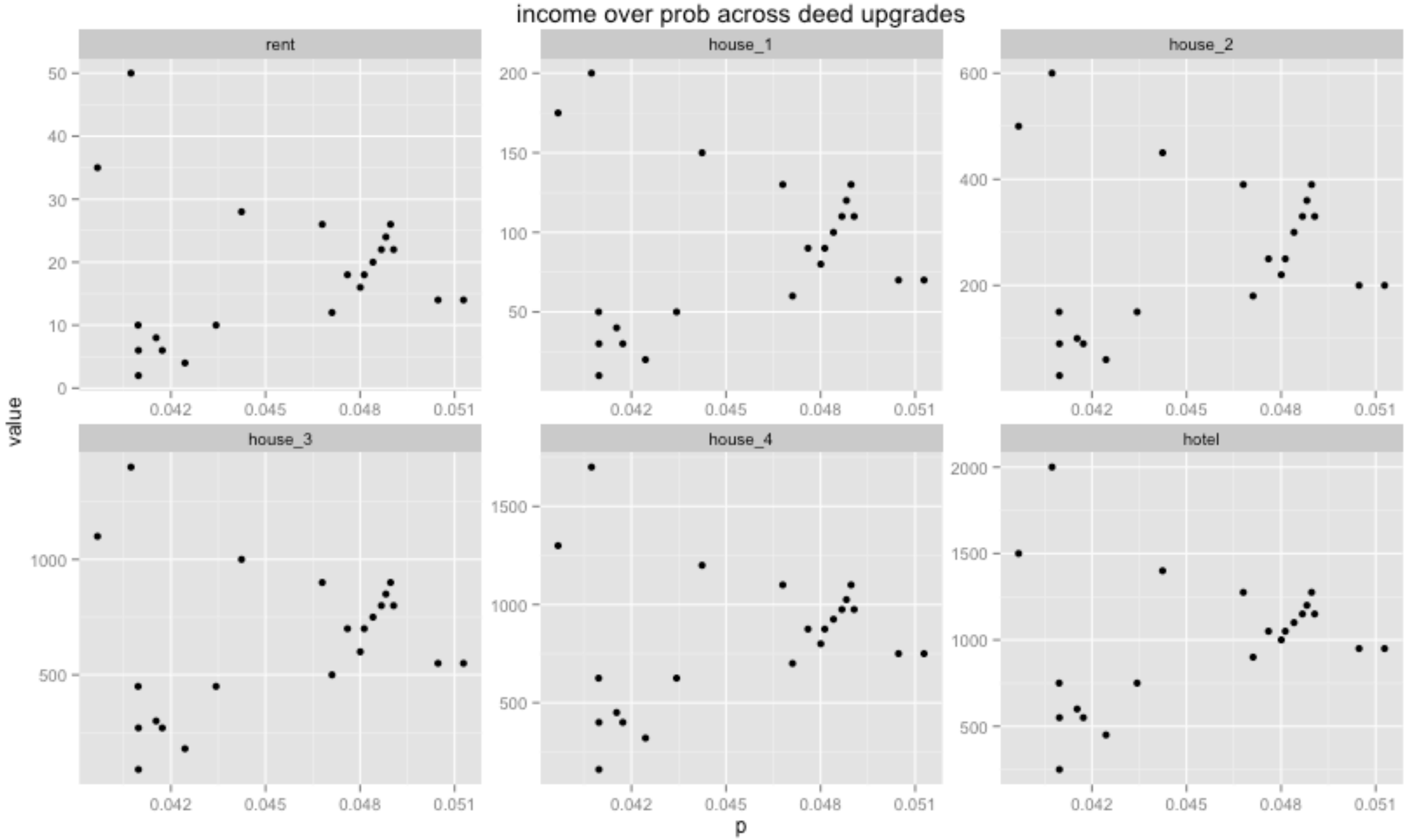
Monopoly Example



Monopoly Example

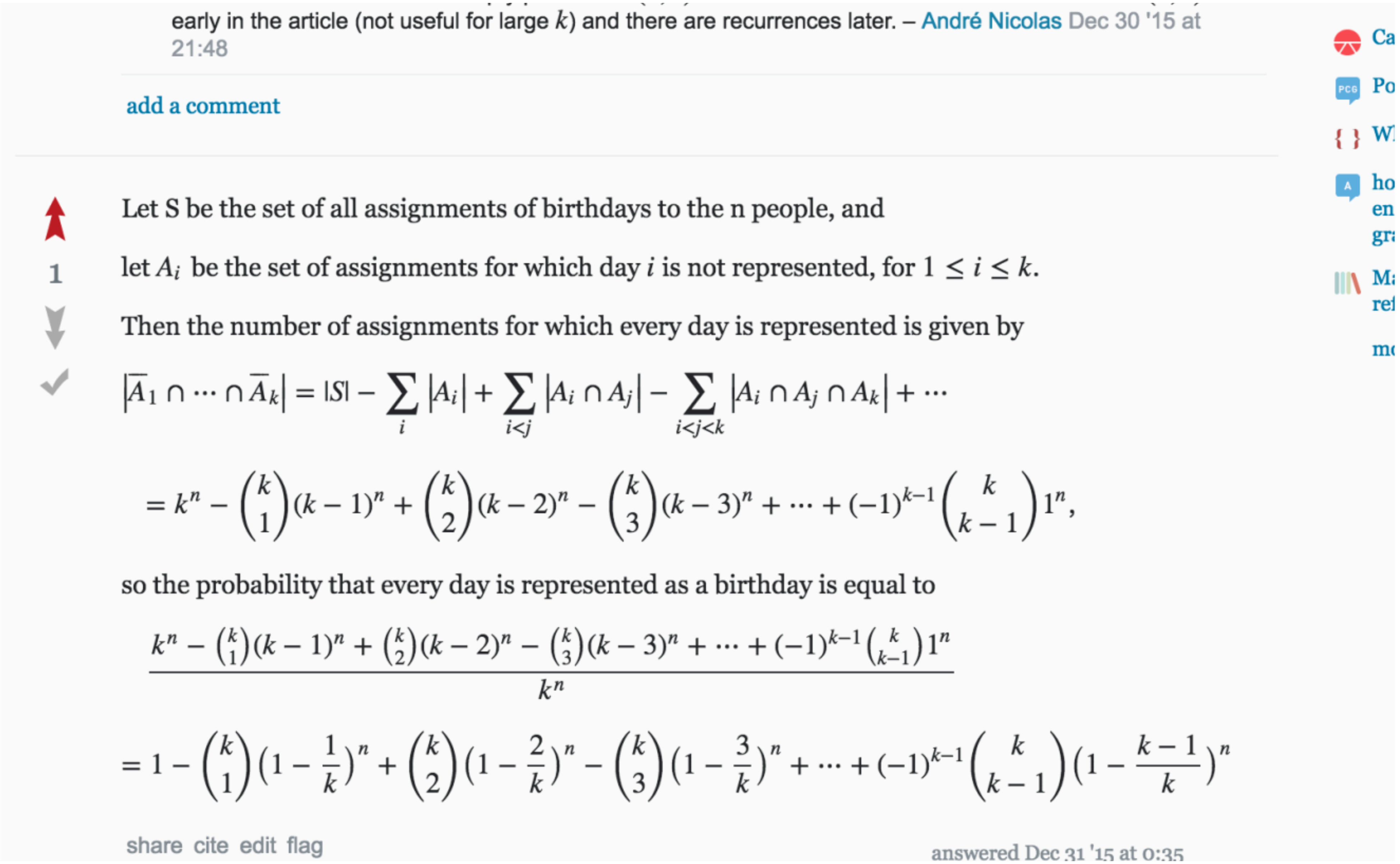


Monopoly Example



Lego Example

Math Overflow?



early in the article (not useful for large k) and there are recurrences later. – André Nicolas Dec 30 '15 at 21:48

add a comment

Let S be the set of all assignments of birthdays to the n people, and

let A_i be the set of assignments for which day i is not represented, for $1 \le i \le k$.

Then the number of assignments for which every day is represented is given by

$$\left|\overline{A}_1 \cap \cdots \cap \overline{A}_k\right| = |S| - \sum_i \left|A_i\right| + \sum_{i<j} \left|A_i \cap A_j\right| - \sum_{i<j<k} \left|A_i \cap A_j \cap A_k\right| + \cdots$$

$$= k^n - \binom{k}{1}(k-1)^n + \binom{k}{2}(k-2)^n - \binom{k}{3}(k-3)^n + \cdots + (-1)^{k-1}\binom{k}{k-1}1^n,$$

so the probability that every day is represented as a birthday is equal to

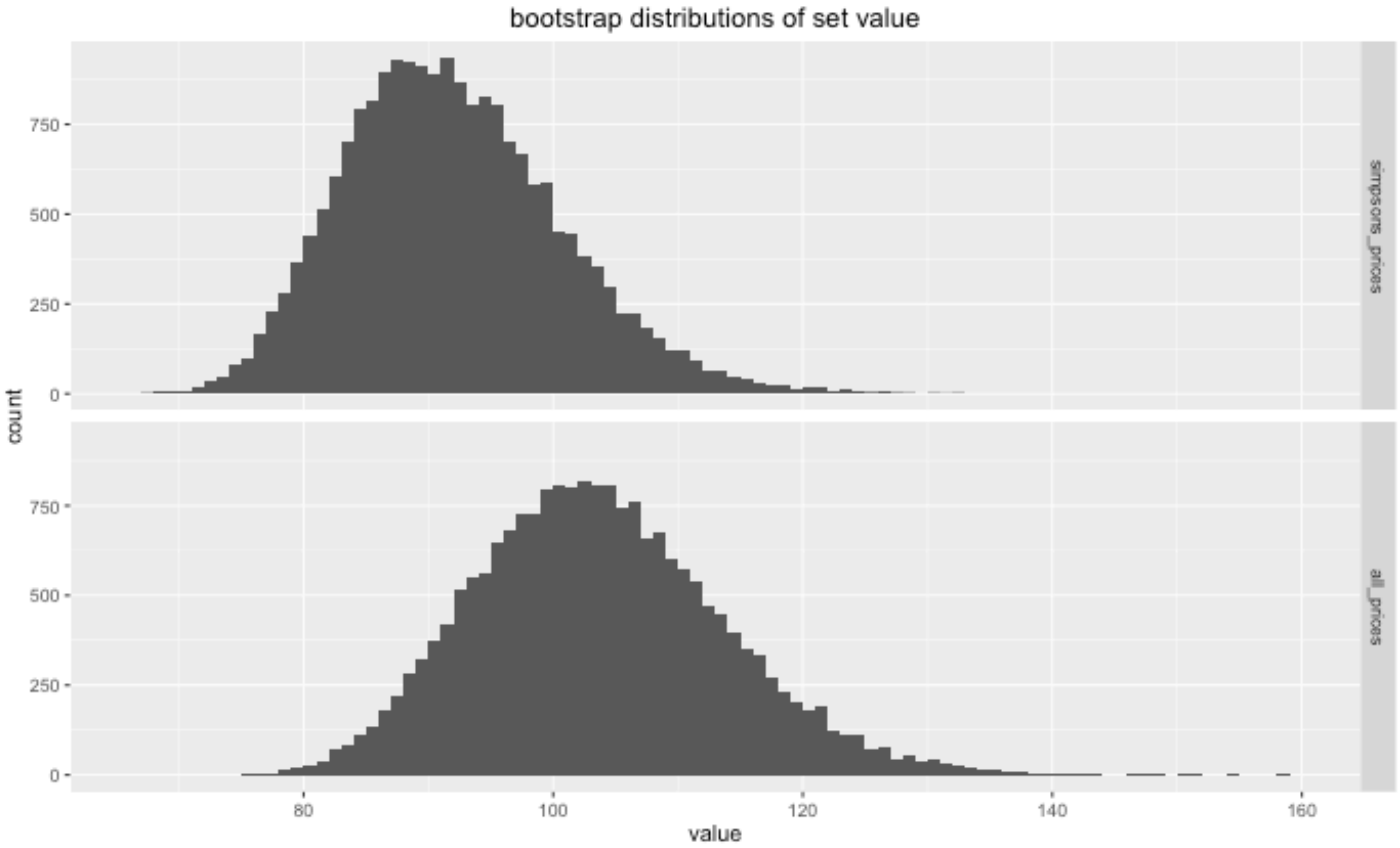
$$\frac{k^n - \binom{k}{1}(k-1)^n + \binom{k}{2}(k-2)^n - \binom{k}{3}(k-3)^n + \cdots + (-1)^{k-1}\binom{k}{k-1}1^n}{k^n}$$

$$= 1 - \binom{k}{1}\left(1 - \frac{1}{k}\right)^n + \binom{k}{2}\left(1 - \frac{2}{k}\right)^n - \binom{k}{3}\left(1 - \frac{3}{k}\right)^n + \cdots + (-1)^{k-1}\binom{k}{k-1}\left(1 - \frac{k-1}{k}\right)^n$$

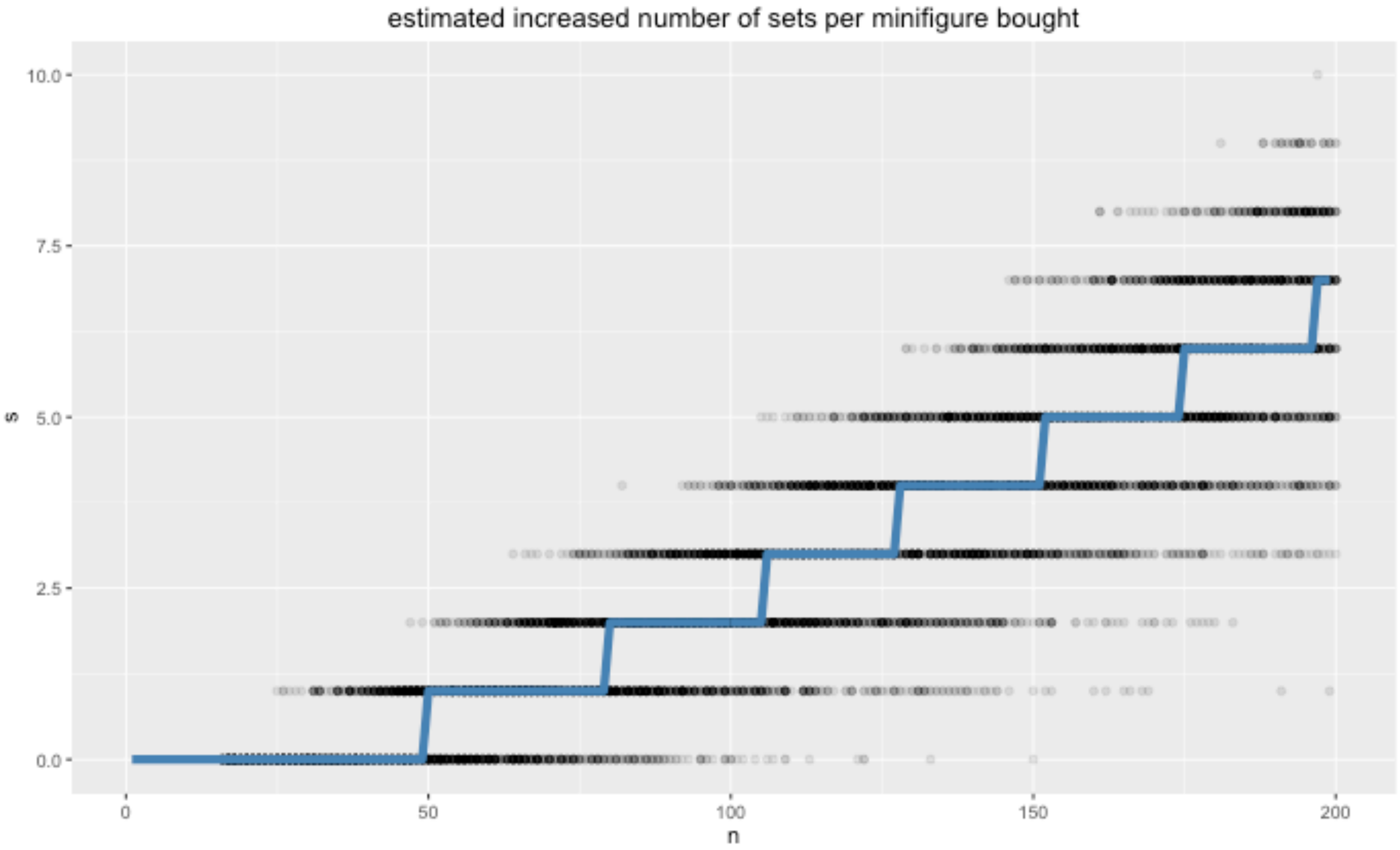
share cite edit flag

answered Dec 31 '15 at 0:35

Lego Example

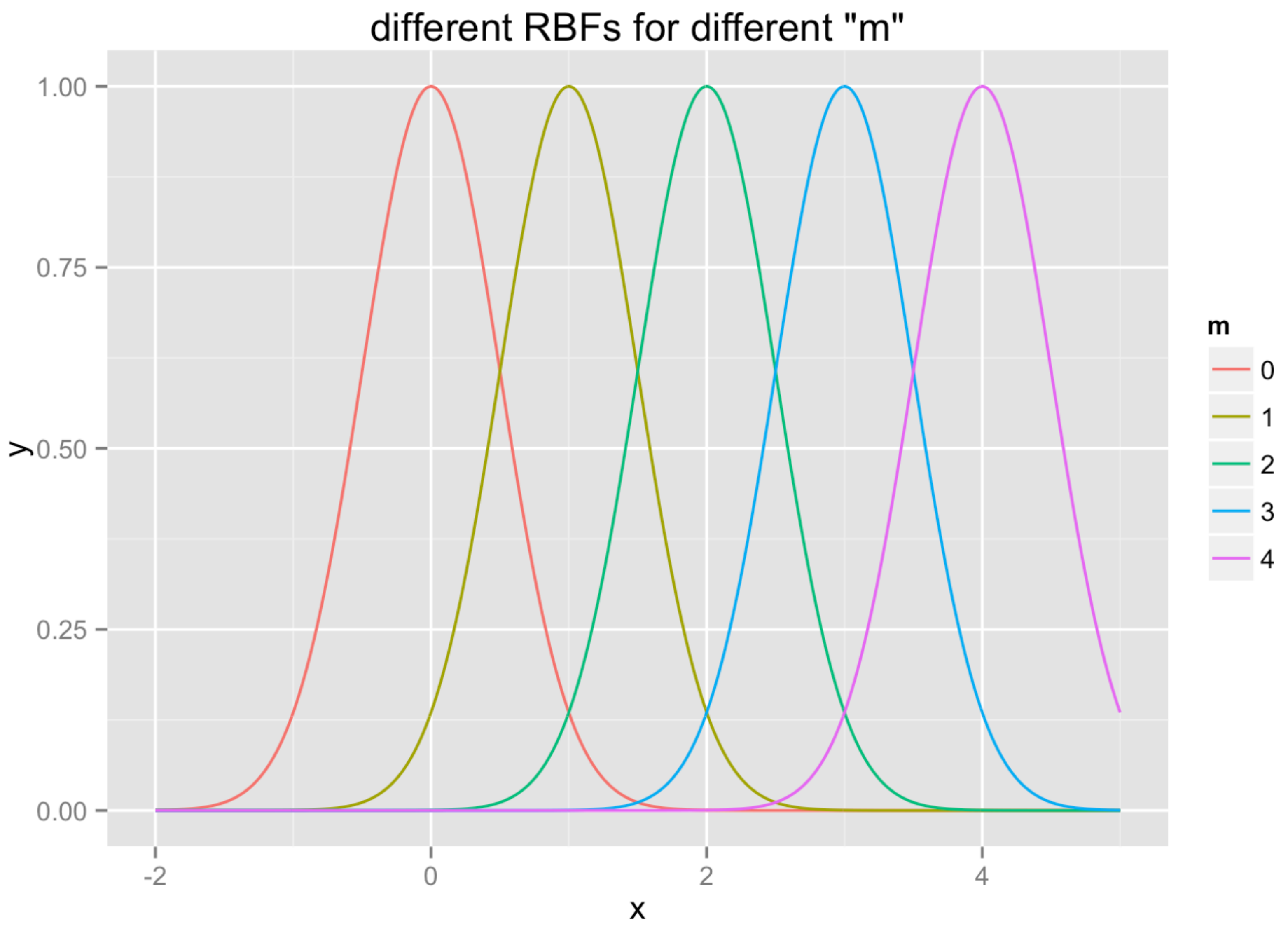


Lego Example

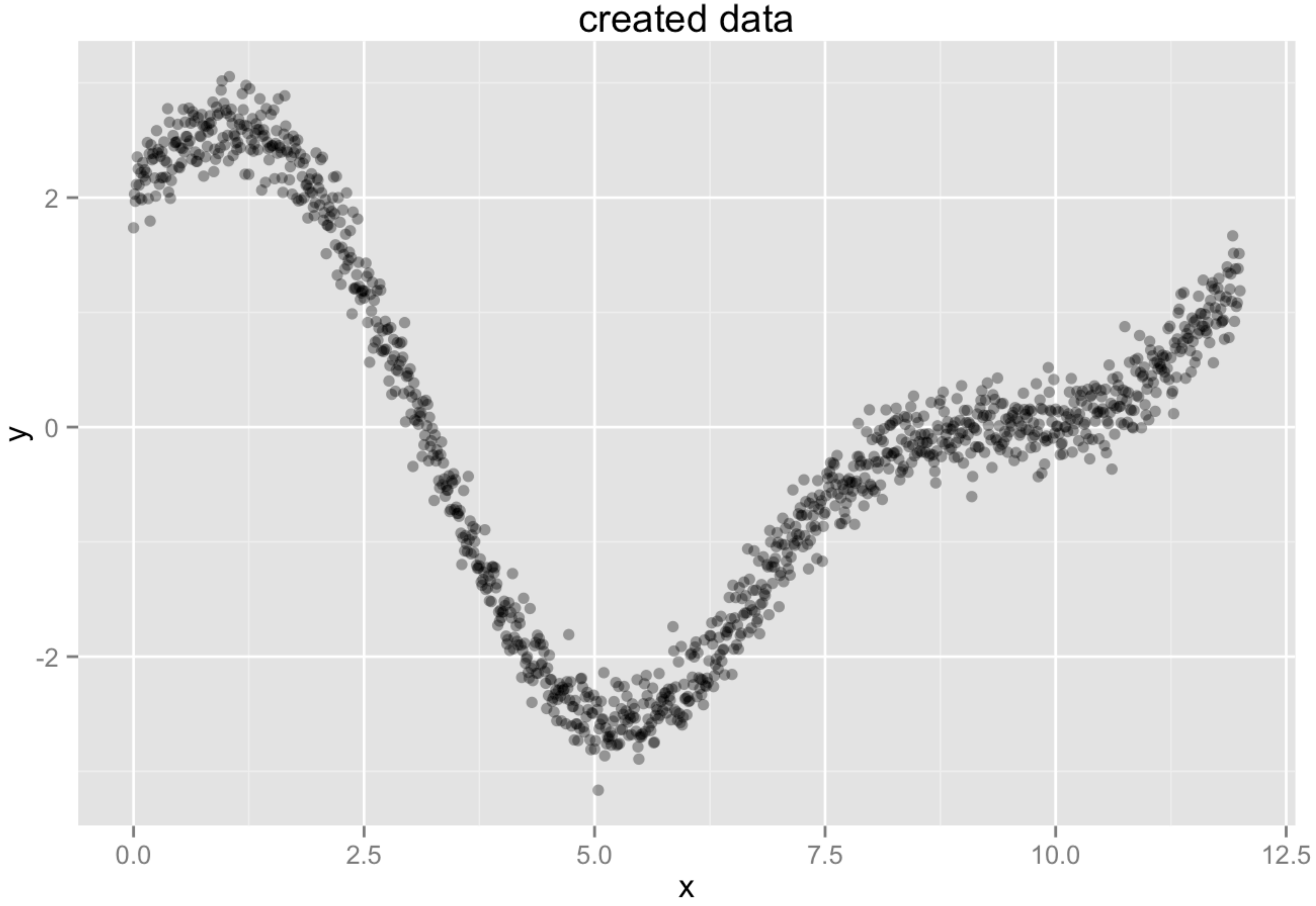


Profit (someday)

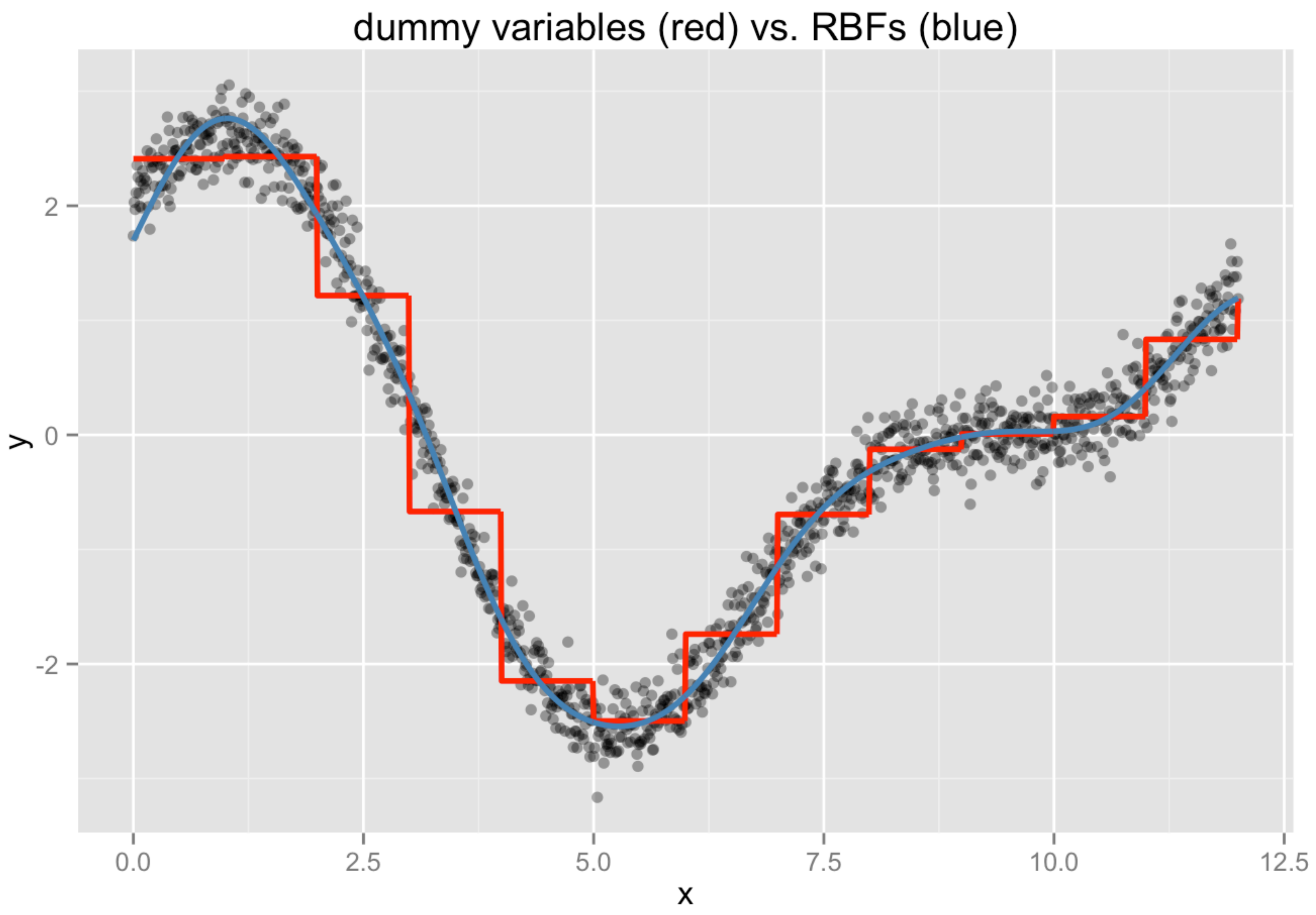
Radial Basis Functions



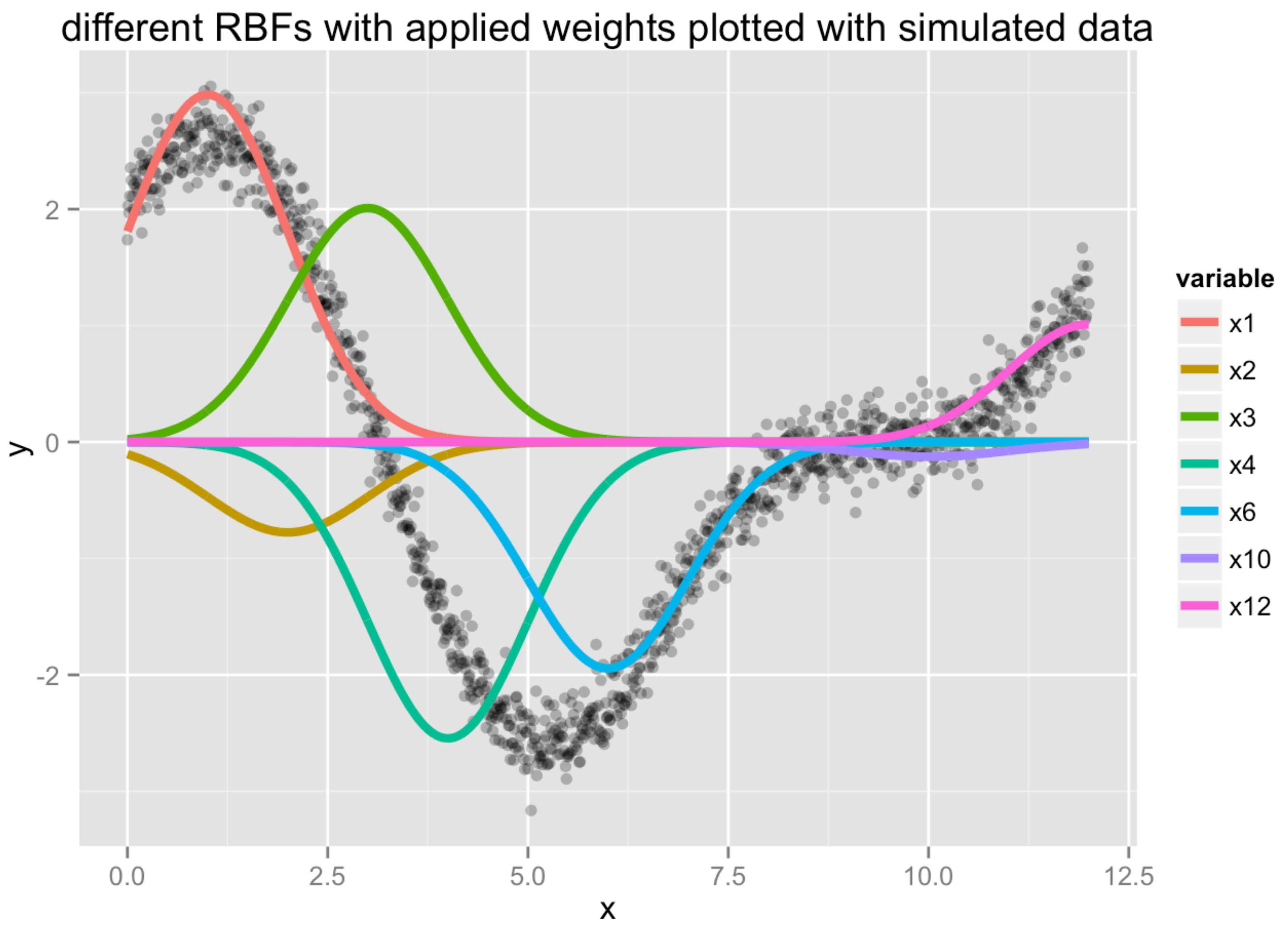
Radial Basis Functions



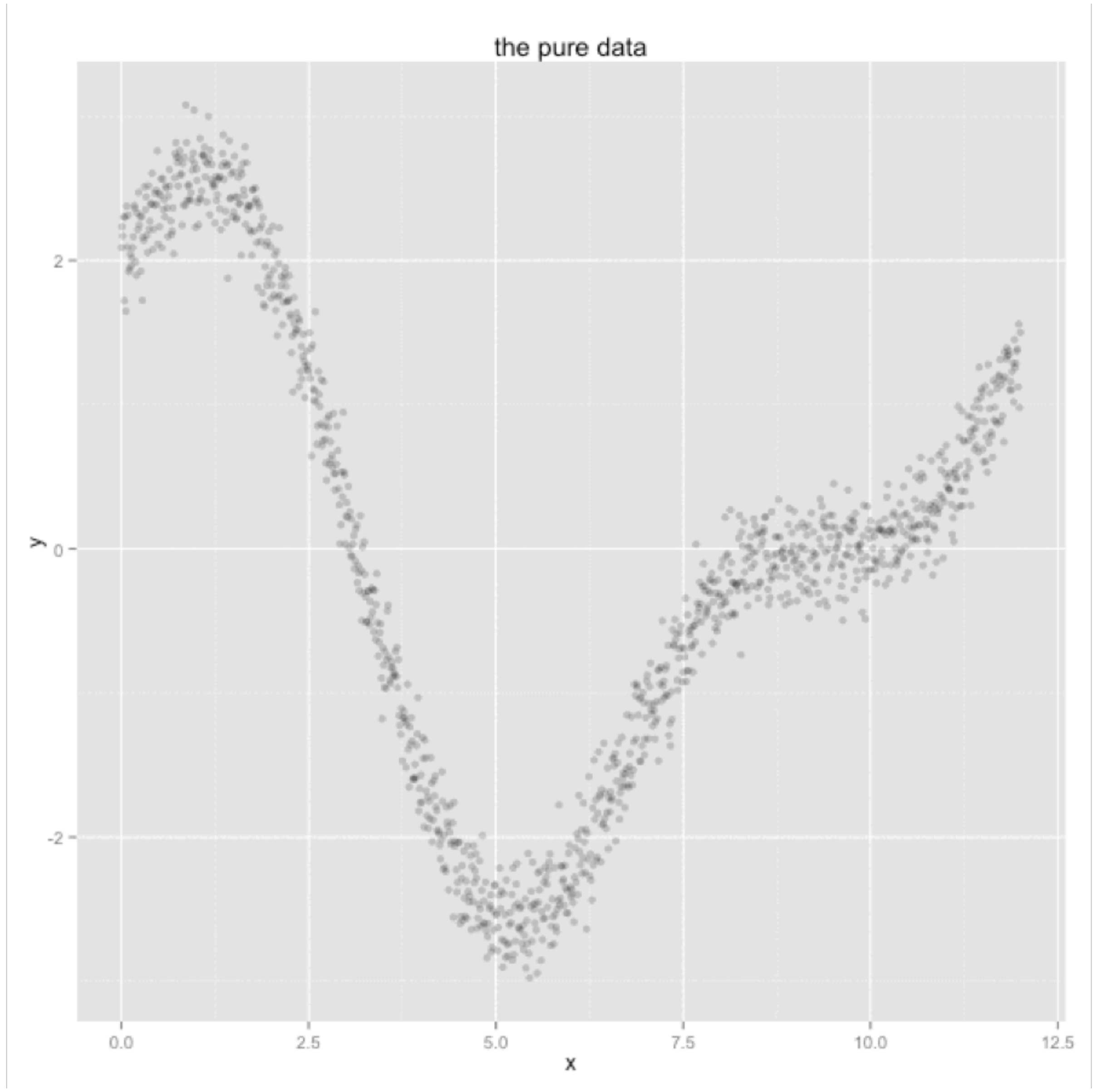
Radial Basis Functions



Radial Basis Functions



Radial Basis Functions



Recap

I was able to omit a lot of words by visualising this properly.

The basic graphs work rather well. With just a scatterplot you can really tell a lot of stories. The gif adds value, but may not be needed.

Let's now discuss some things that make this a nice visualisation.

Proper Viz

Some things about these past examples;

- some of them are standalone, some of them still need explaining
- the visualisations seem to have similar base styles
- the style feels a little bit minimal, but it is able to get a point across clearly

The next slide contains a gif to explain this.

Remove

to improve

(the **data-ink** ratio)

Created by **Darkhorse Analytics** www.**darkhorseanalytics**.com

Remove

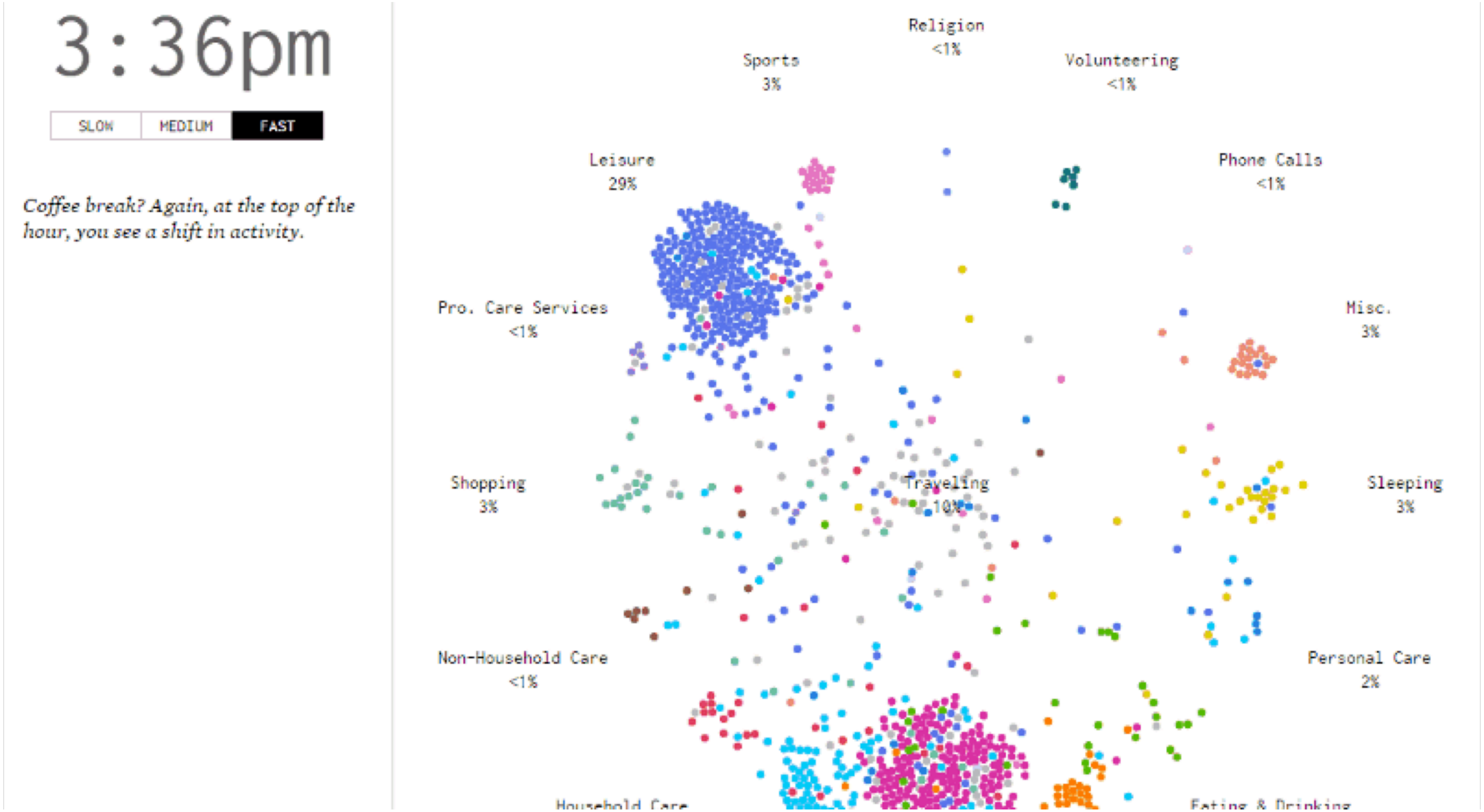
to improve

the **pie chart** edition

Created by Darkhorse Analytics

www.**darkhorseanalytics**.com

Let's look at something gone



3:36pm

Coffee break? Again, at the top of the hour, you see a shift in activity.

Goal

In day to day work, you want to make decisions or understand something. If your visualisation doesn't adhere to this, it is an art project not a professional tool. I want science, not art.

Next Up

We've now talked about the end result, let's now talk about tools and principles on how to make these.

Grammars

To explain how we'll make these viz we'll work with a small example. You'll notice I use the R language. Why?

> *"R is not a DSL. It’s a language for writing DSLs, which is something altogether more powerful"*
>
> – @jcheng

Farmer Fred

We're going to do an ABCD test based on different diets for different chickens.

We'll pretend to be consultants, except that we're obviously much better than most McKinsey people because we can actually code.

We'll use the dataset `ChickWeight` that comes with R.

We have data.

Now what?

We have data.

Why not just look at it?

Guess.

What does this code do?

```
> head(ChickWeight)
```

Guess.

What does this code do?

```
> head(ChickWeight)
  weight Time Chick Diet
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
```

Guess.

What does this code do?

```
> summary(ChickWeight)
```

Guess.

What does this code do?

```
> summary(ChickWeight)
     weight           Time           Chick     Diet
 Min.   : 35.0   Min.   : 0.00   13     : 12   1:220
 1st Qu.: 63.0   1st Qu.: 4.00   9      : 12   2:120
 Median :103.0   Median :10.00   20     : 12   3:120
 Mean   :121.8   Mean   :10.72   10     : 12   4:118
 3rd Qu.:163.8   3rd Qu.:16.00   17     : 12
 Max.   :373.0   Max.   :21.00   19     : 12
                                 (Other):506
```

Feeling.

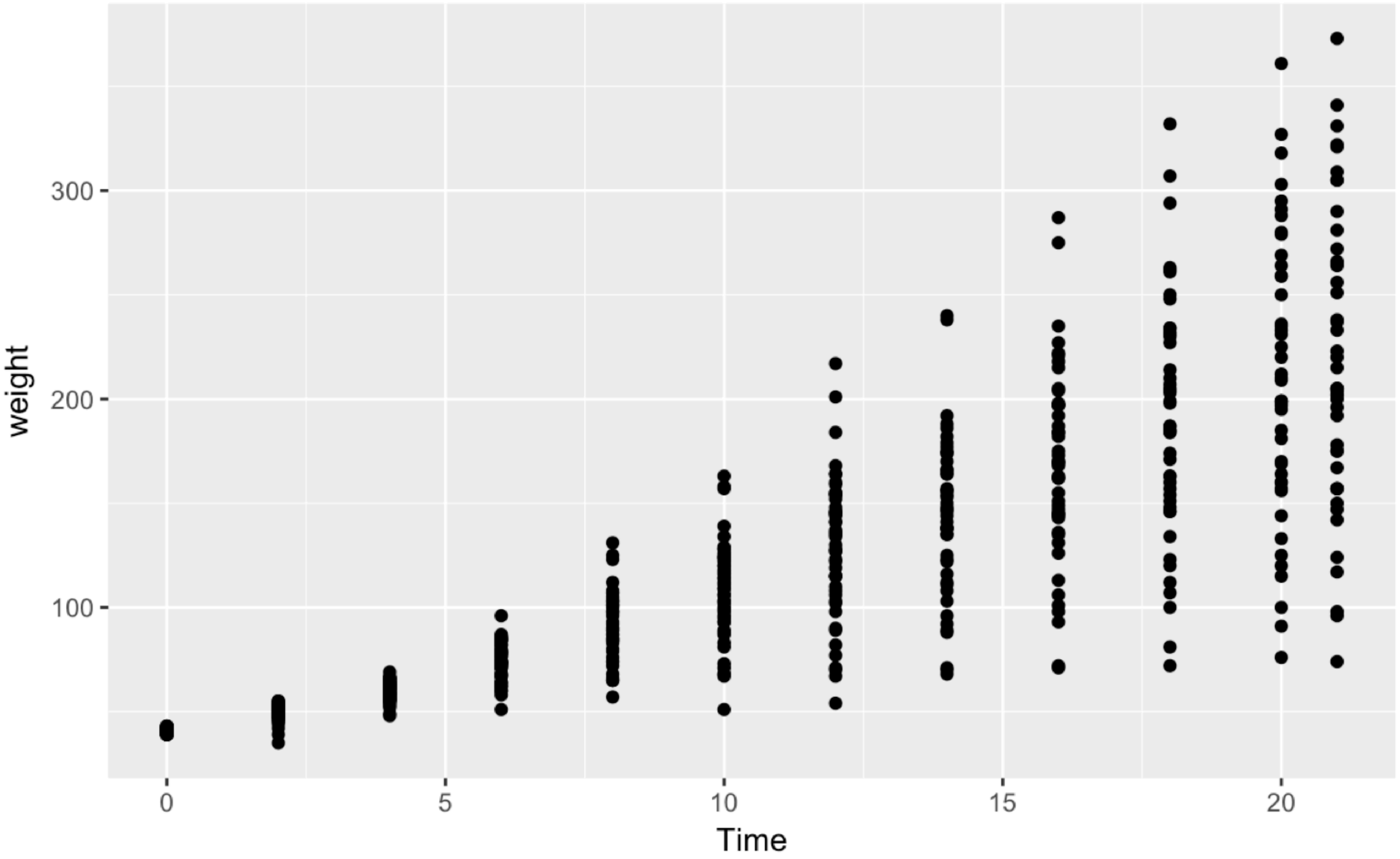
We now know what we can expect from the data.

Let's now actually look at it by visualising it.

Guess

What does this code do?

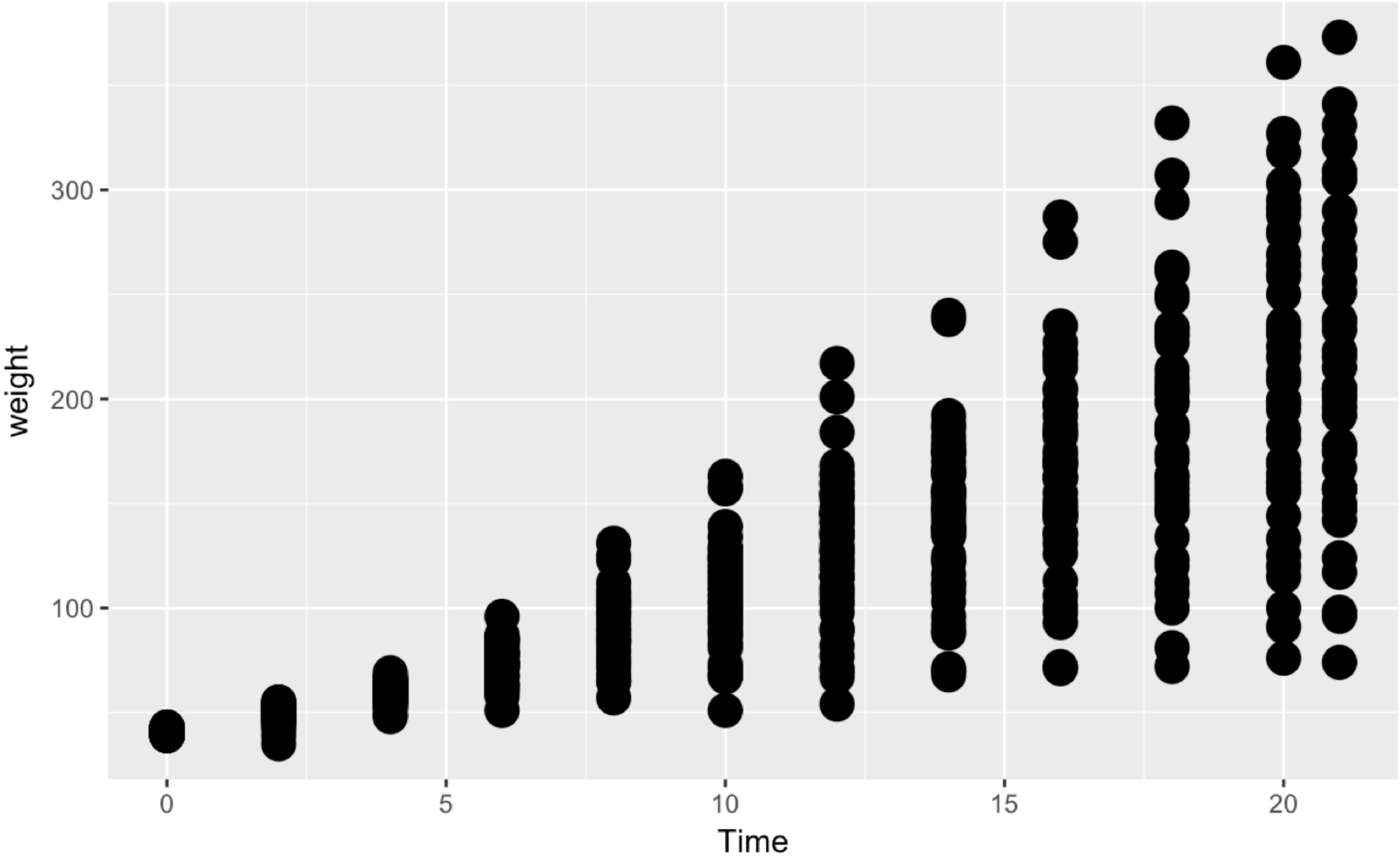
```
p <- ggplot()
p + geom_point(
  data=ChickWeight,
  aes(x=Time, y=weight)
)
```


weight
300
200
100
0
5
10
15
20
Time

Guess

What does this code do?

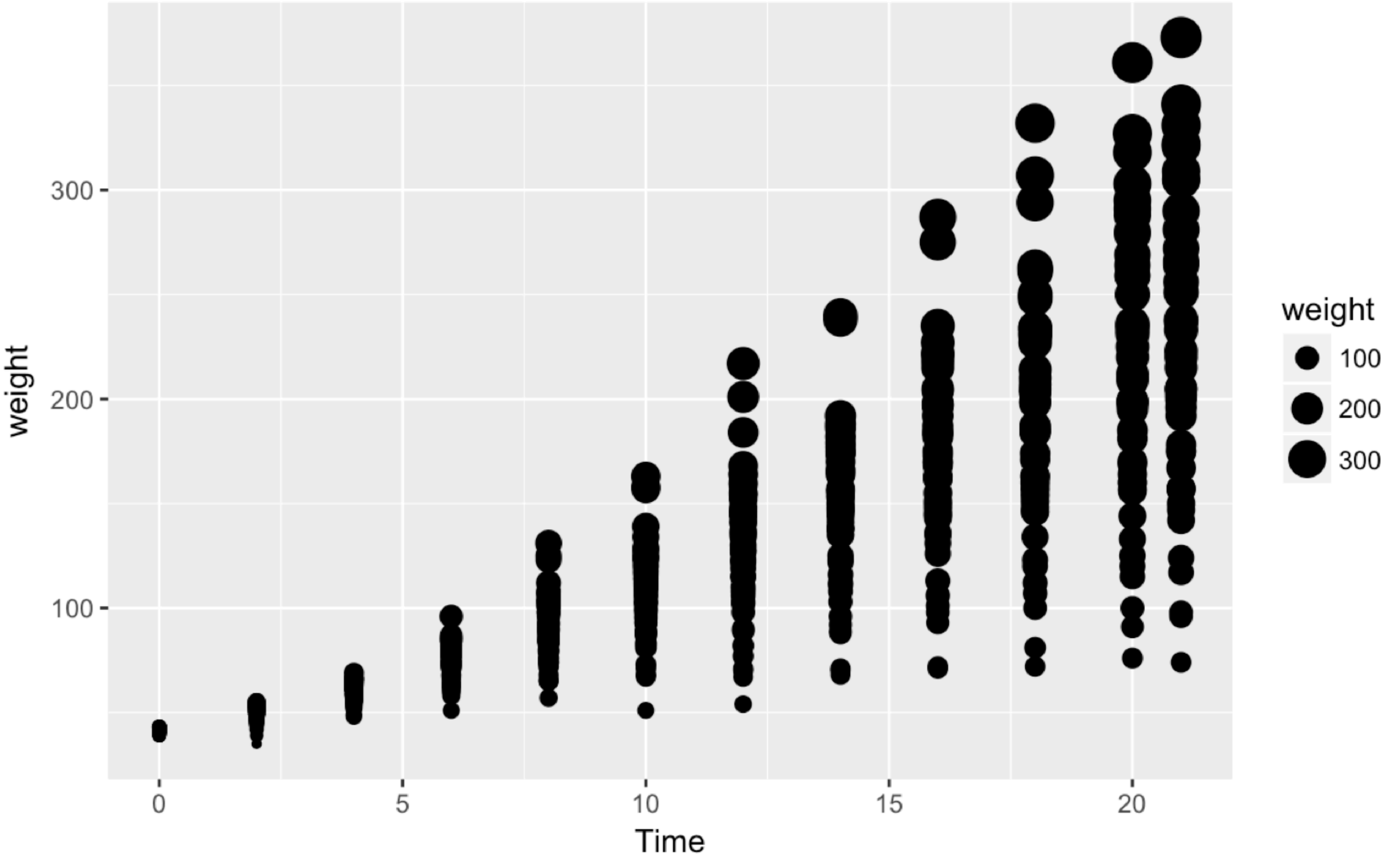
```
p <- ggplot()
p + geom_point(
  data=ChickWeight,
  aes(x=Time, y=weight),
  size = 5
)
```


weight
300
200
100
0
5
10
15
20
Time

Guess

What does this code do?

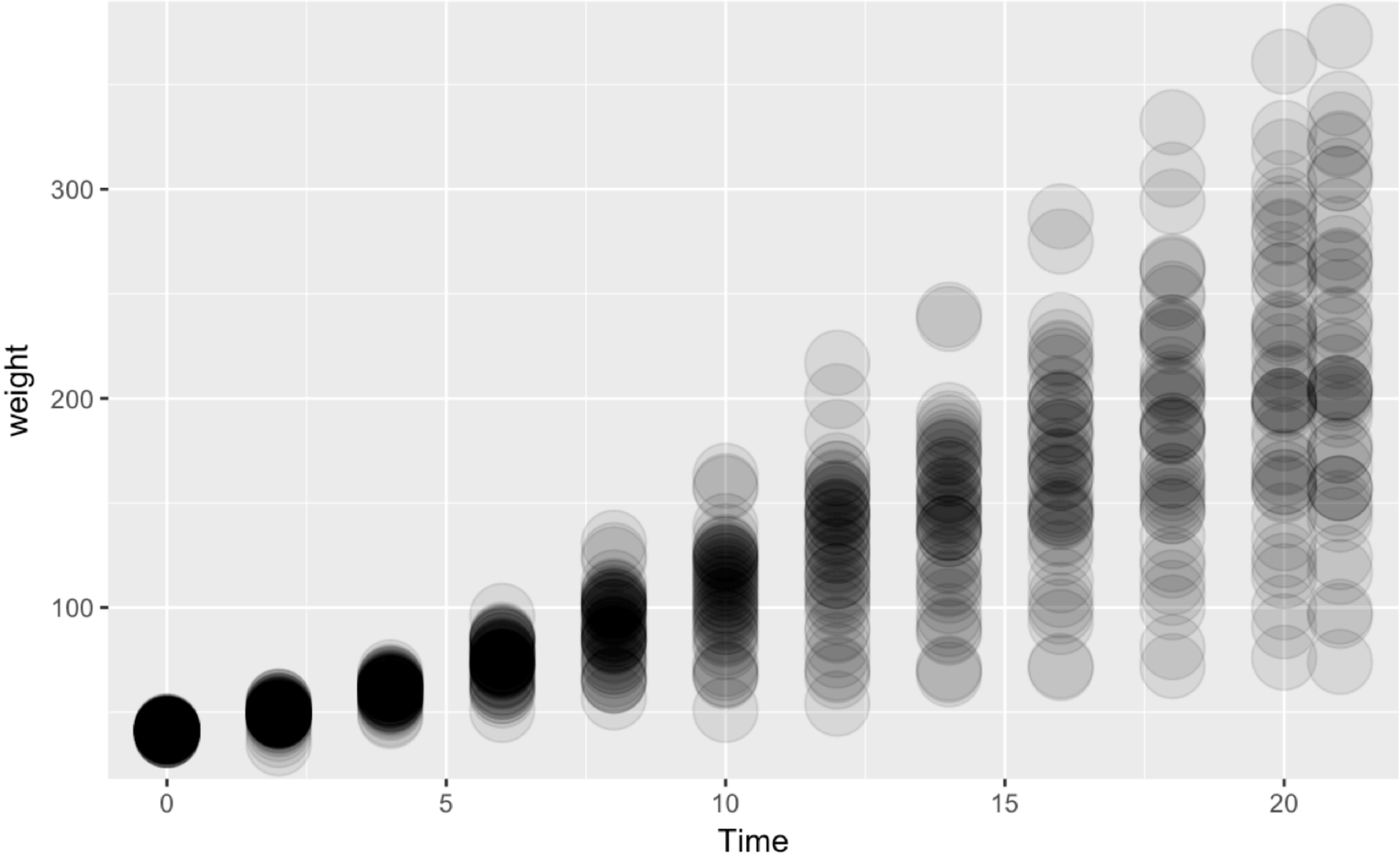
```
p <- ggplot()
p + geom_point(
  data=ChickWeight,
  aes(x=Time, y=weight, size = weight)
)
```


weight
300
200
100
0
5
10
15
20
Time
weight
● 100
● 200
● 300

Guess

What does this code do?

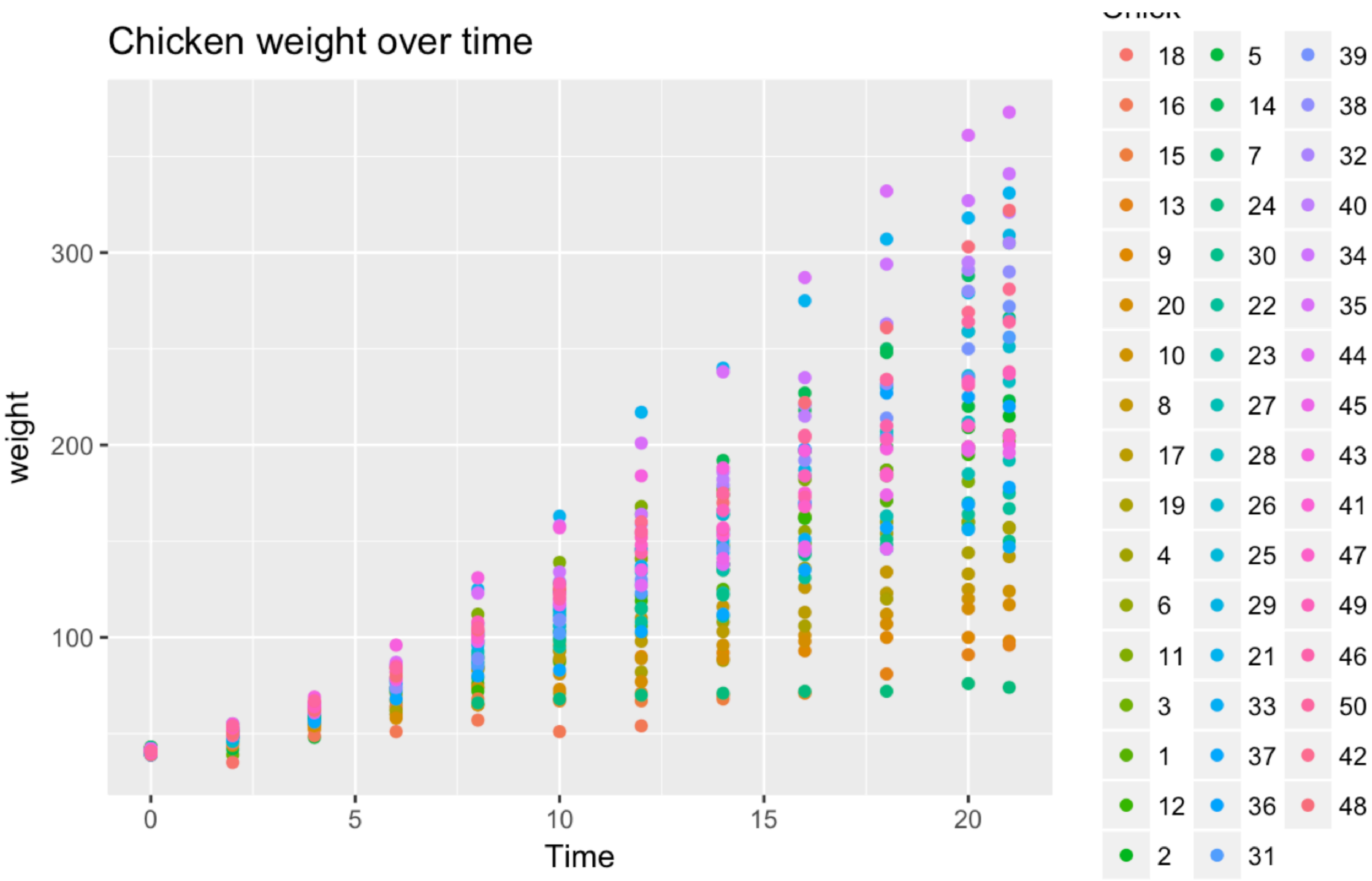
```
p <- ggplot()
p + geom_point(
  data=ChickWeight,
  aes(x=Time, y=weight),
  size = 10, alpha = 0.1
)
```


weight
300
200
100
0
5
10
15
20
Time

Guess

What does this code do?

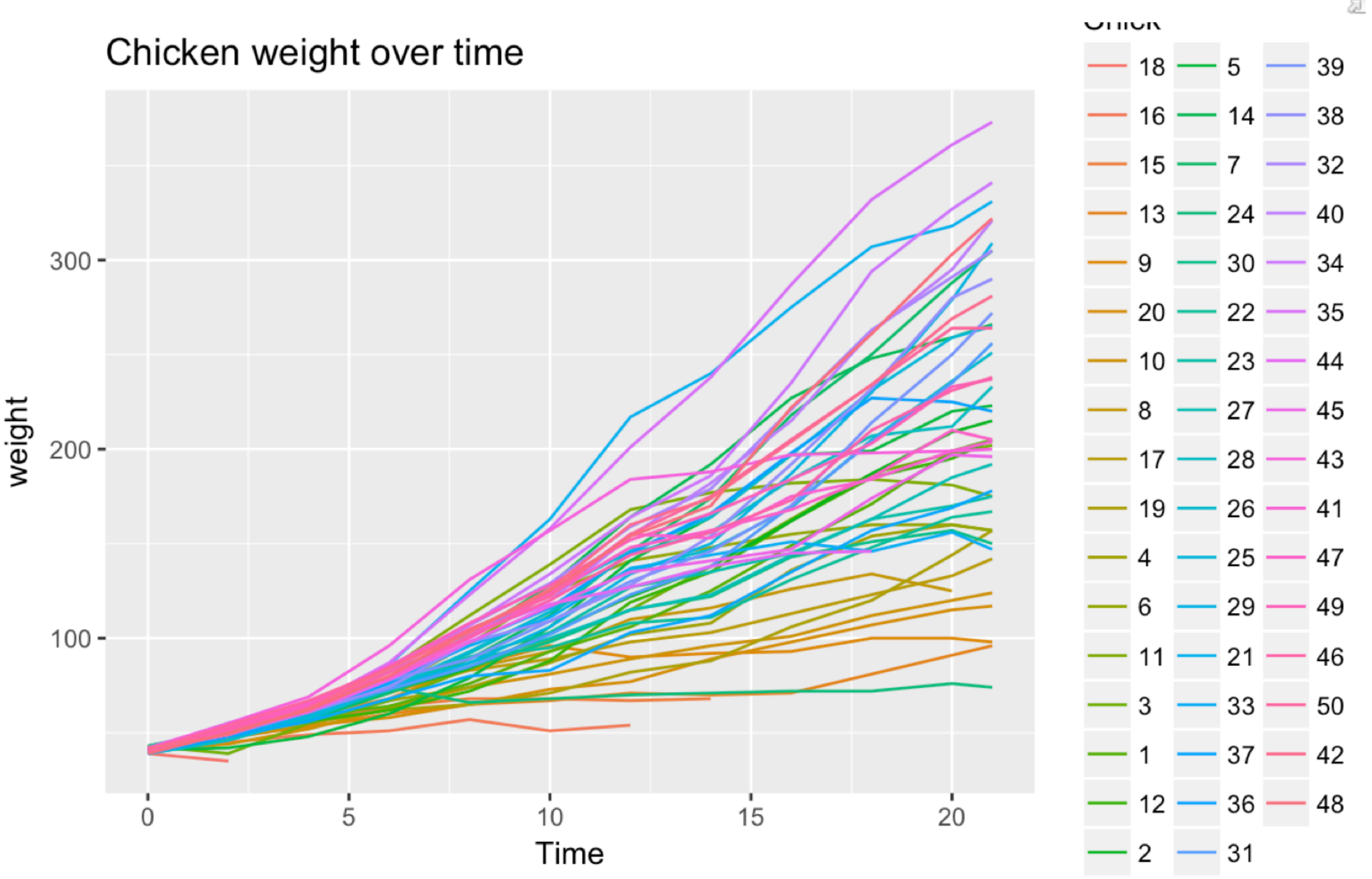
```
p <- ggplot()
p <- p + geom_point(
  data=ChickWeight,
  aes(x=Time, y=weight, colour=Chick)
)
p + ggtitle("Chicken weight over time")
```


Chick
Chicken weight over time
weight
300
200
100
0
5
10
15
20
Time
18
16
15
13
9
20
10
8
17
19
4
6
11
3
1
12
2
5
14
7
24
30
22
23
27
28
26
25
29
21
33
37
36
31
39
38
32
40
34
35
44
45
43
41
47
49
46
50
42
48

Guess

What does this code do?

```
p <- ggplot()
p <- p + geom_line(
  data=ChickWeight,
  aes(x=Time, y=weight, colour=Chick)
)
p + ggtitle("Chicken weight over time")
```

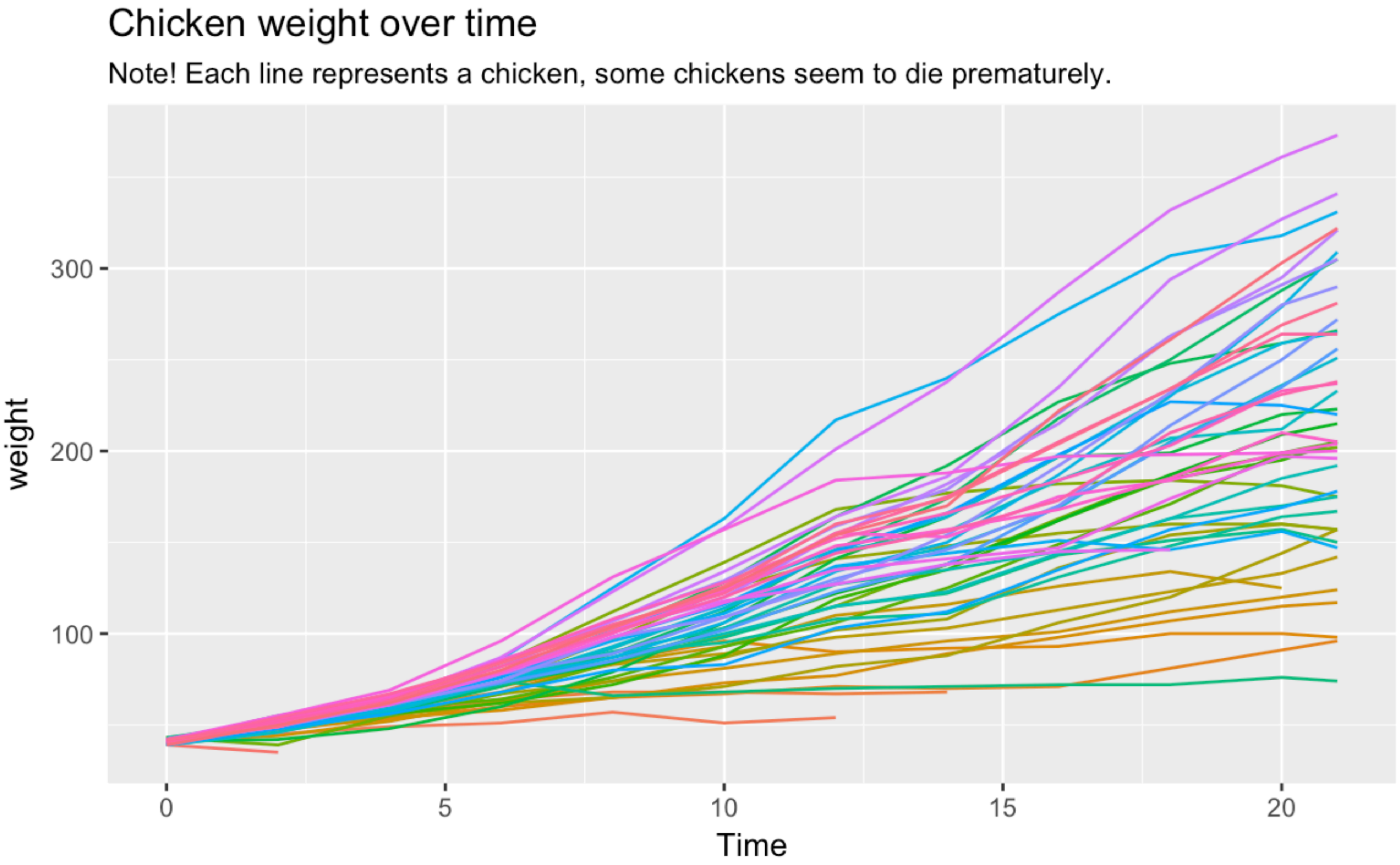

Chicken weight over time
weight
300
200
100
0
5
10
15
20
Time
Chick
18
16
15
13
9
20
10
8
17
19
4
6
11
3
1
12
2
5
14
7
24
30
22
23
27
28
26
25
29
21
33
37
36
31
39
38
32
40
34
35
44
45
43
41
47
49
46
50
42
48

Guess

What does this code do?

```
p <- ggplot() +
    geom_line(
        data=ChickWeight,
        aes(x=Time, y=weight, colour=Chick)) +
    theme(legend.position="none")

p + ggtitle("Chicken weight over time",
    subtitle = "Note! Some chickens seem to die prematurely!")
```



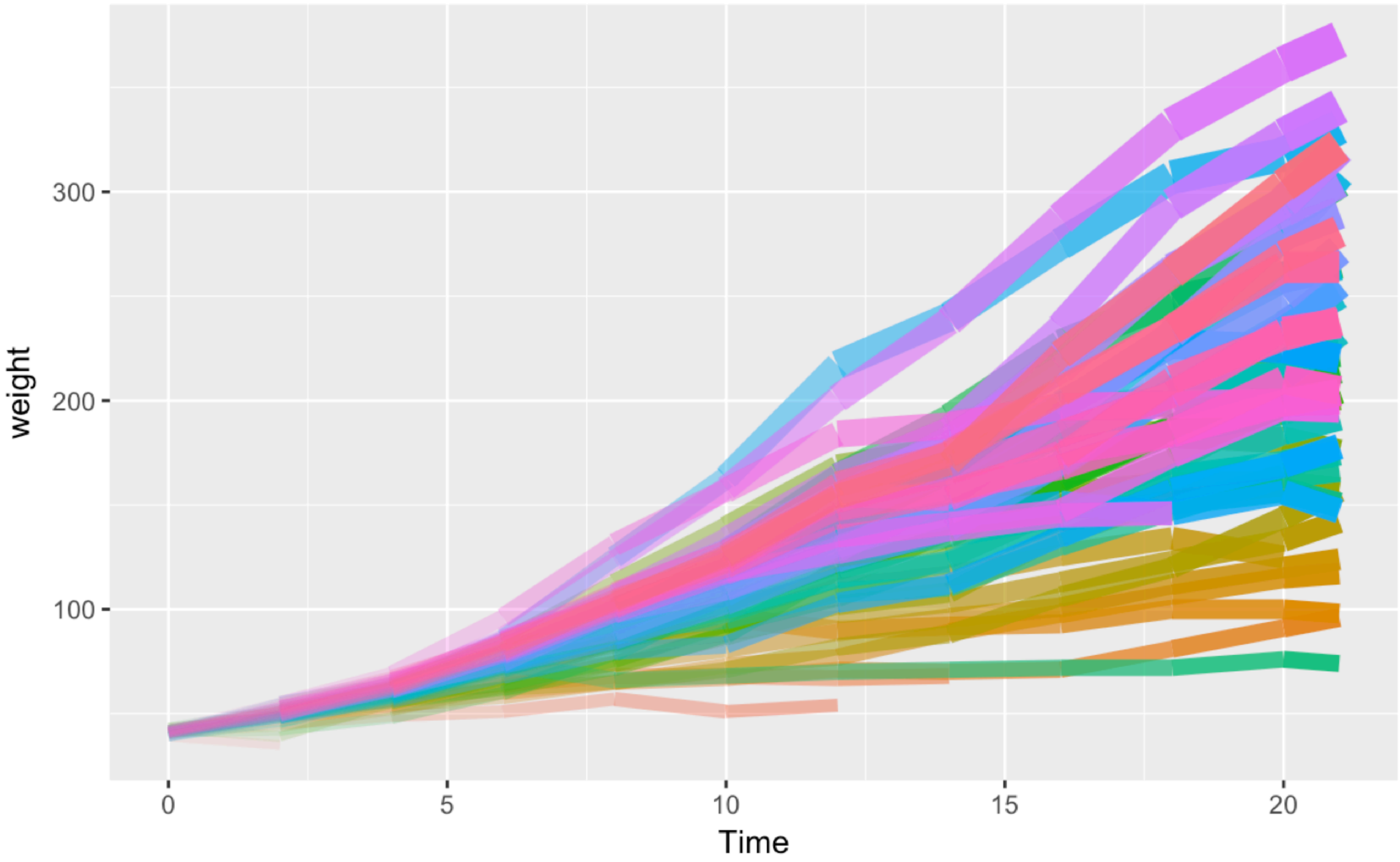
Chicken weight over time

Note! Each line represents a chicken, some chickens seem to die prematurely.

Guess

What does this code do?

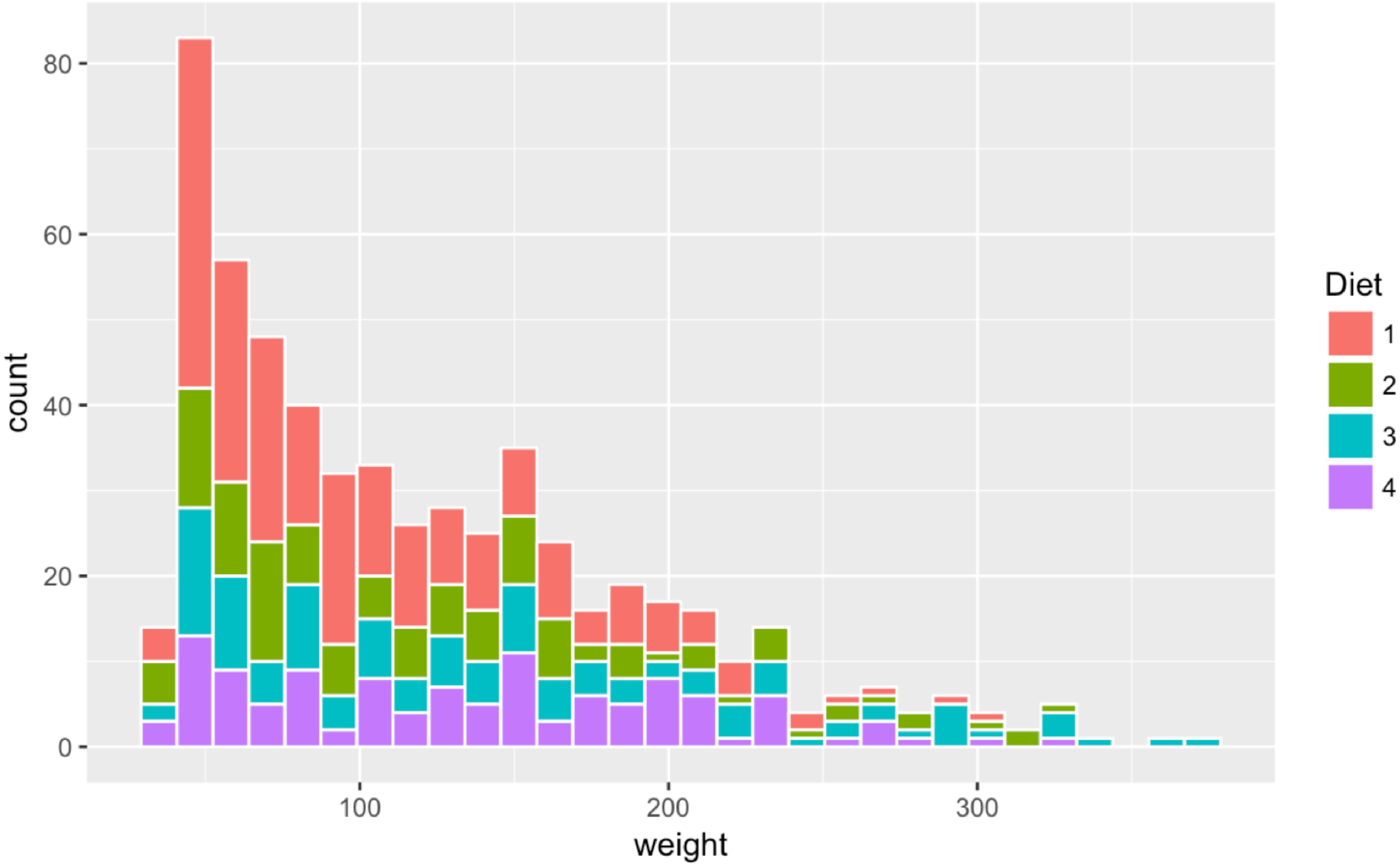
```
ggplot() +
  geom_line(data=ChickWeight,
            aes(x=Time, y=weight,
                colour=Chick, alpha=Time,
                size=weight)) +
  theme(legend.position="none")
```


weight
300
200
100
0
5
10
15
20
Time

Guess

What does this code do?

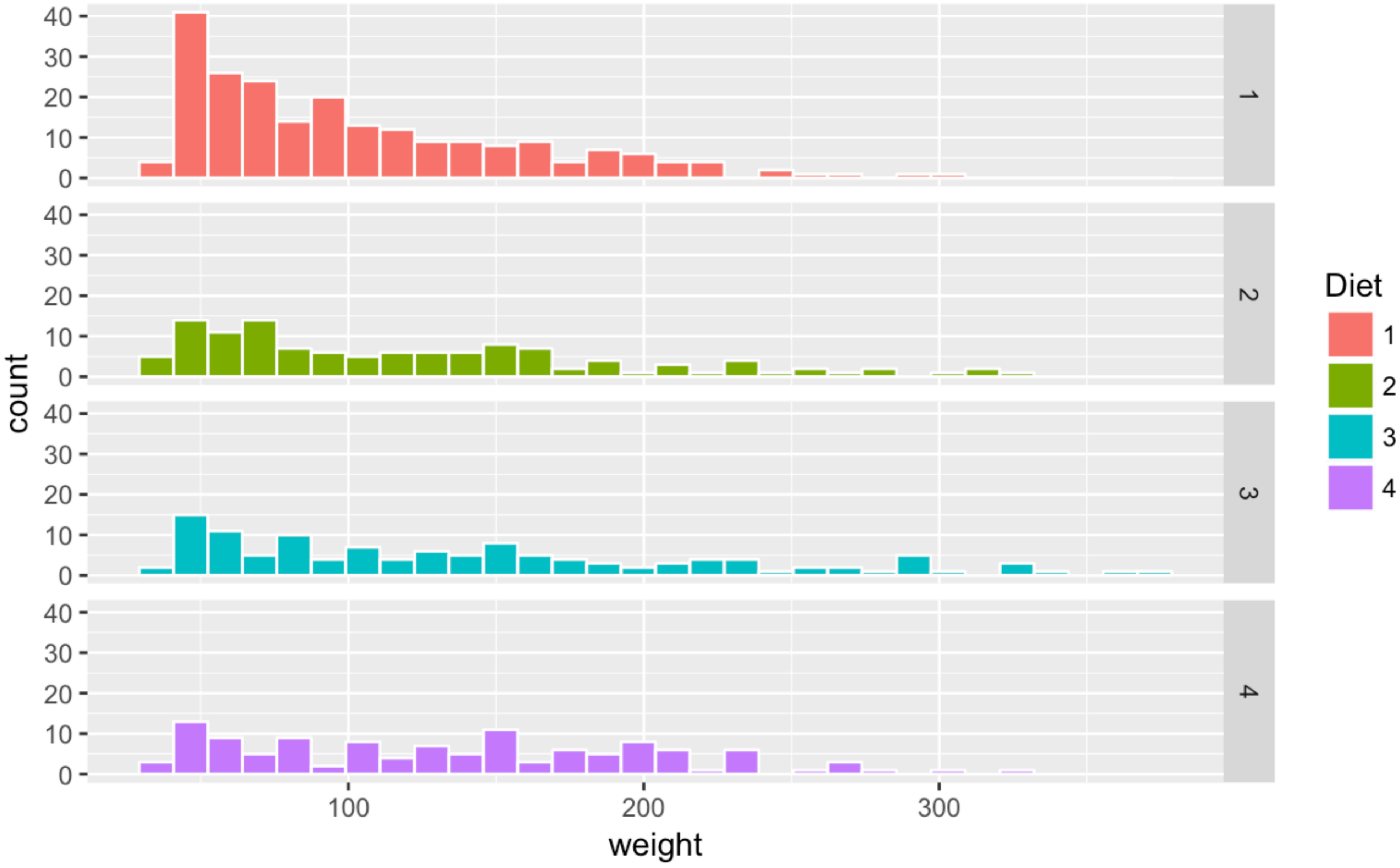
```
p <- ggplot()
p + geom_histogram(
  data=ChickWeight,
  aes(x=weight, fill=Diet),
  colour="white"
)
```


80
60
40
20
0
count
100
200
300
weight
Diet
1
2
3
4

Guess

What does this code do?

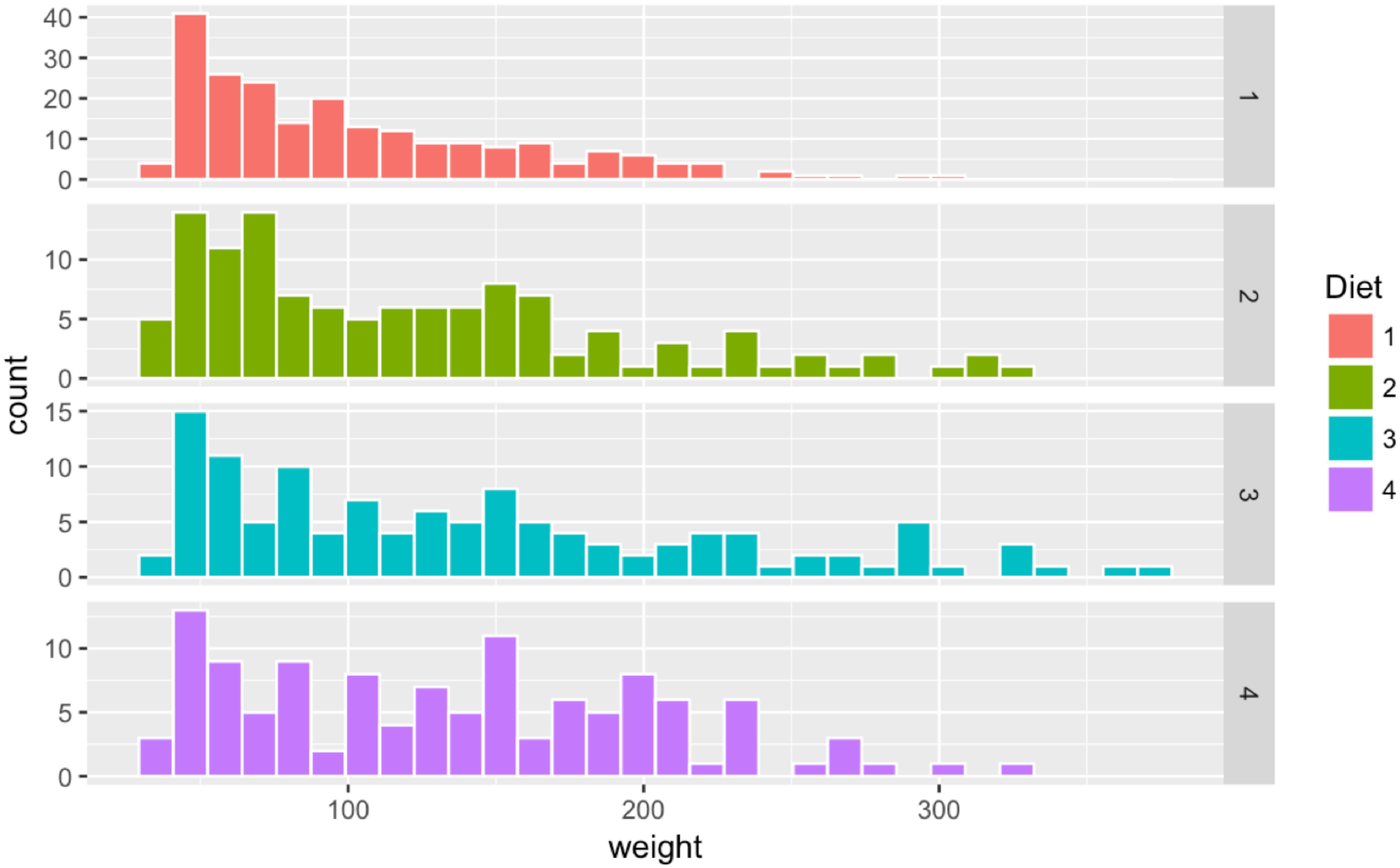
```
p <- ggplot()
p <- p + geom_histogram(
  data=ChickWeight,
  aes(x=weight, fill=Diet),
  colour="white"
)
p + facet_grid(Diet ~ .)
```


40
30
20
10
0
count
1
2
3
4
100
200
300
weight
Diet
1
2
3
4

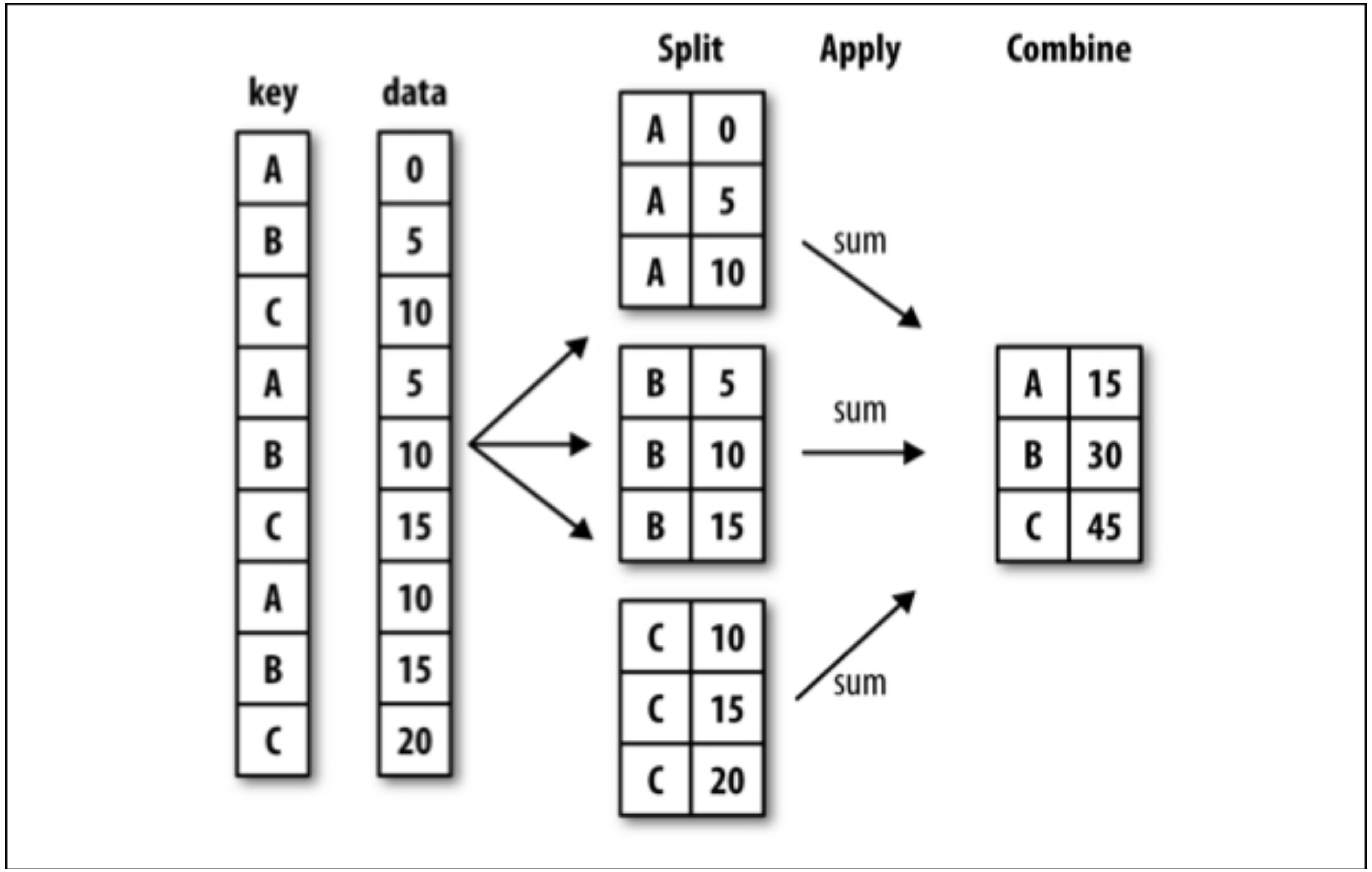
Guess

What does this code do?

```
p <- ggplot()
p <- p + geom_histogram(
  data=ChickWeight,
  aes(x=weight, fill=Diet),
  colour="white"
)
p + facet_grid(Diet ~ ., scales = "free_y")
```


40
30
20
10
0
10
5
0
15
10
5
0
10
5
0
count
1
2
3
4
100
200
300
weight
Diet
1
2
3
4

Split Apply Combine



key	data
A	0
B	5
C	10
A	5
B	10
C	15
A	10
B	15
C	20

Split

A	0
A	5
A	10

B	5
B	10
B	15

C	10
C	15
C	20

Combine

A	15
B	30
C	45

Grammar of Data Manipulation

What might this code do?

```
ChickWeight %>%
  group_by(Diet) %>%
  summarise(m = mean(weight))
```

Grammar of Data Manipulation

Notice the joy.

```
ChickWeight %>%
  group_by(Diet, Time) %>%
  summarise(m = mean(weight))
```

Grammar of Data Manipulation

Notice the joy.

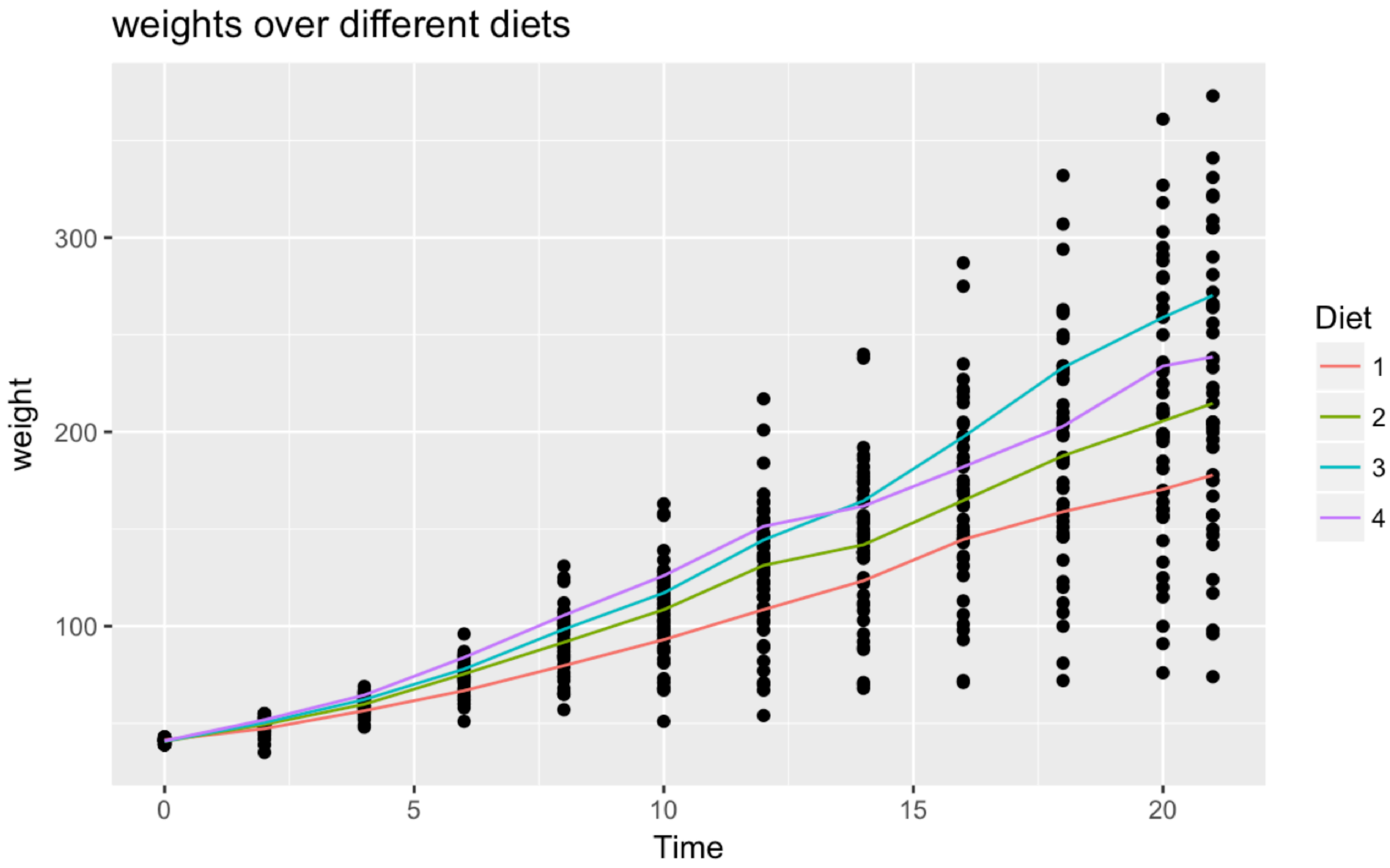
```
ChickWeight %>%
  filter(Chick != 10) %>%
  group_by(Diet, Time) %>%
  summarise(m = mean(weight),
            v = var(weight))
```

Note that this grammar is just a UI. The backend can be R, a database or even spark.

Guess what does this code do?

```
agg <- ChickWeight %>%
  group_by(Diet, Time) %>%
  summarise(m = mean(weight))

ggplot() +
  geom_point(data=ChickWeight, aes(Time, weight)) +
  geom_line(data=agg, aes(Time, m, colour=Diet)) +
  ggtitle("weights over different diets")
```


weights over different diets
weight
300
200
100
Time
0
5
10
15
20
Diet
1
2
3
4

Recap Grammar

- having a grammar makes code more understandable
- having a grammar makes analysis more planable
- having a grammar makes visualisation simpler

If you think within the context of a proper grammar it is easy to regard a visualisation as if it is lego. You only need a few good verbs to explain what you want from the data and whatever tool you use needs to facilitate this.

Interactivity

There are some examples where you need more freedom than a static visualisation. These moments are rare but they do exists.

Doing things interactively makes things more complex so please think before you act.

I'll end with three examples, the last one will be the most complex.

Mercator Maps

link

Ensemble Yourself

link

Inverse Turing Test

link

Machine Learning as a User Interface

link

link

Ending thoughts

For 90% of my viz, I use grammars from R. Even when I use python.

For interactive things; be careful. Look at c3, maybe d3.

Seperate concerns and think about abstraction. You only need to search for the simplest possible way to explain things.

We're only communicating after all.