

Lesson 3:

Logistic Regression

This Lesson's Goals

Learn about logistic regression

Make a figure for data from a logistic regression

Do a logistic regression in R

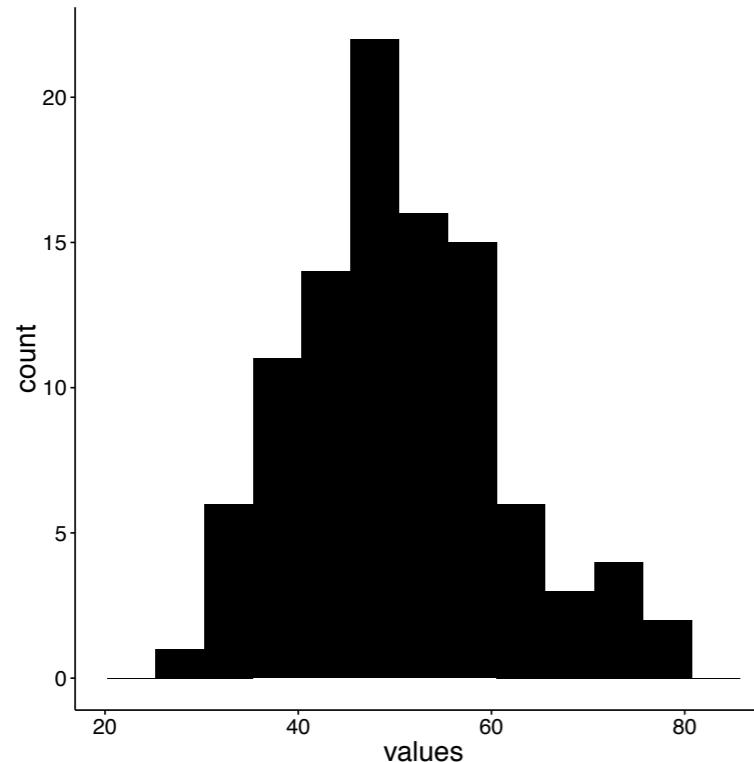
Summarise results in an R Markdown document

Math

linear regression

predict
continuous variables

talk about in regards to
mean and standard deviation

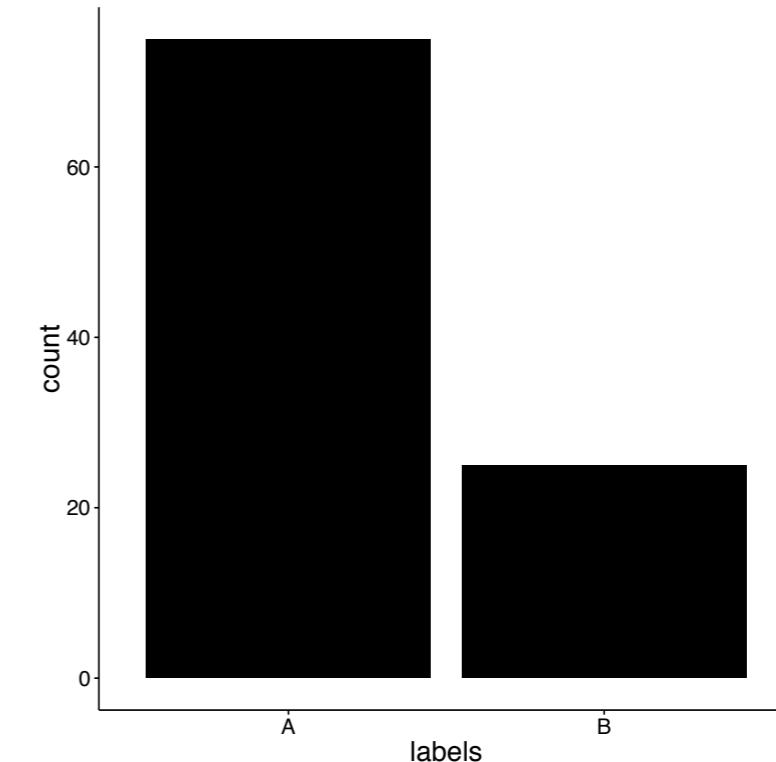


predict specific y-value
given specific x-value

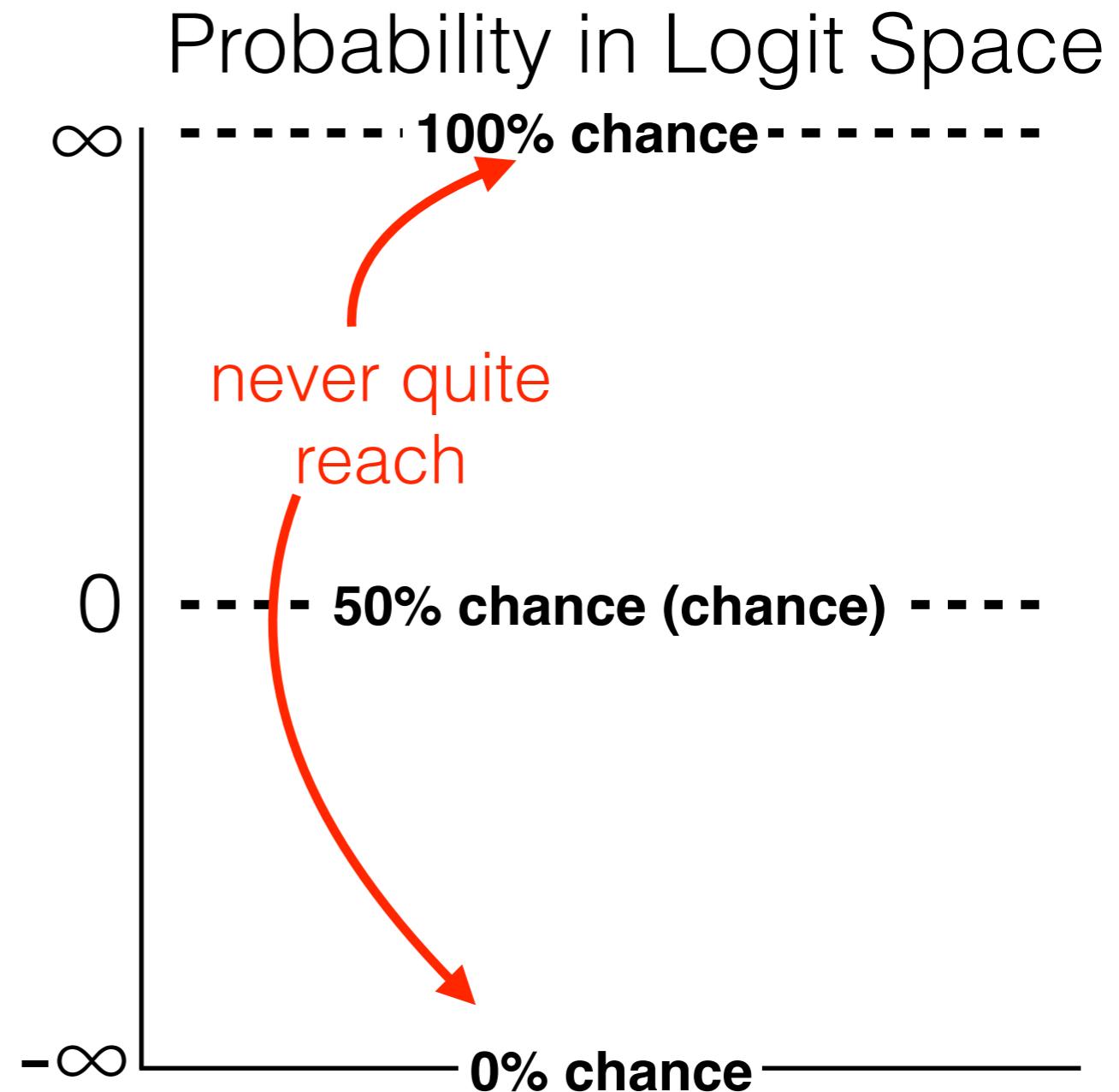
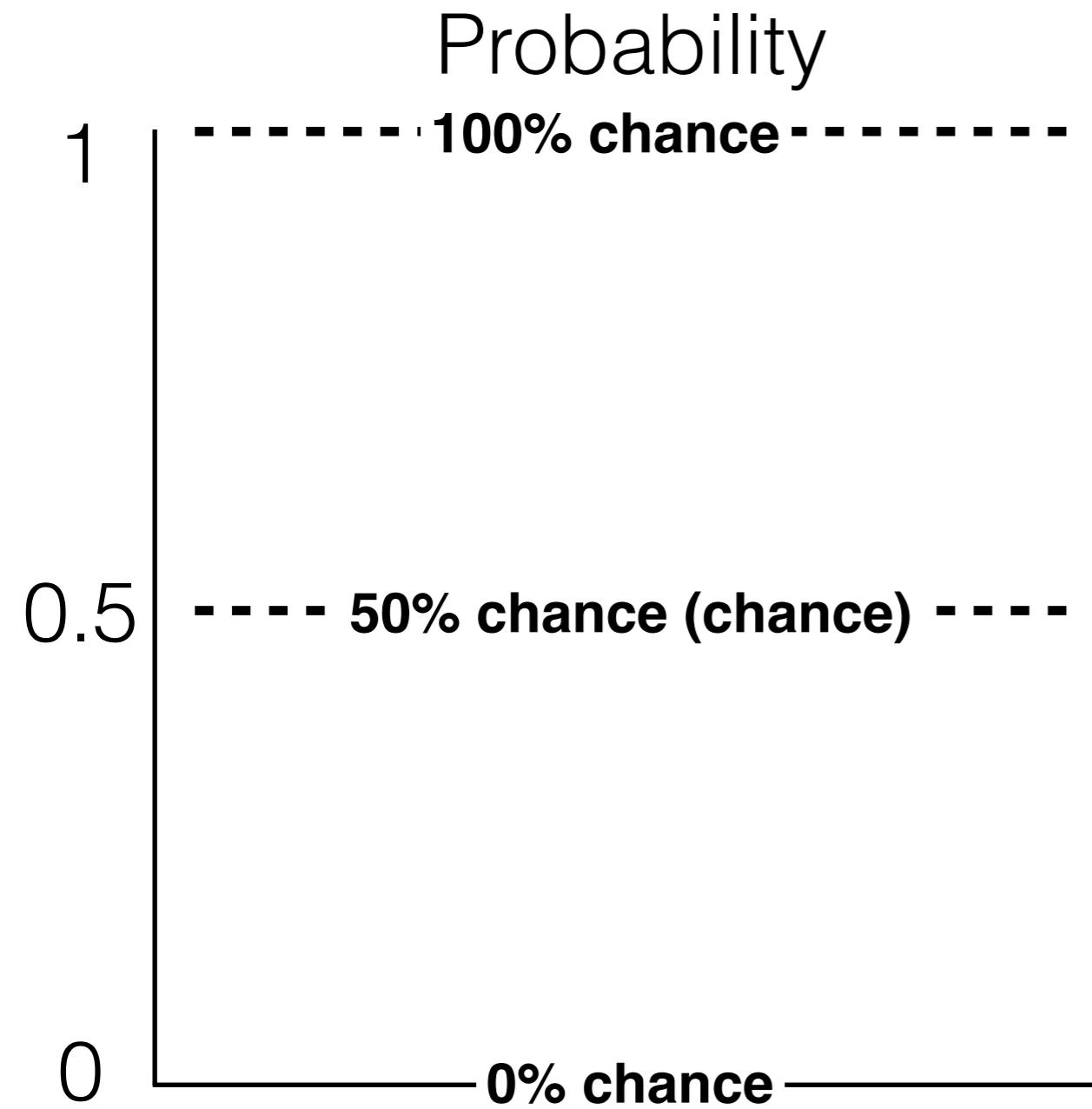
logistic regression

predict
categorical variables

talk about in regards to
counts



predict **probability** y-level
given specific x-value



$$y_i = a + bx_i + e_i$$

$$\text{logit } p_i = a + bx_i$$

$$\log[p/(1-p)]_i = a + bx_i?$$



log odds

no error term

$$\log[p/(1-p)]_i = a + bx_i$$

$\log[p/(1-p)]_i$ = probability of specific y-level (F or T)
(dependent variable)

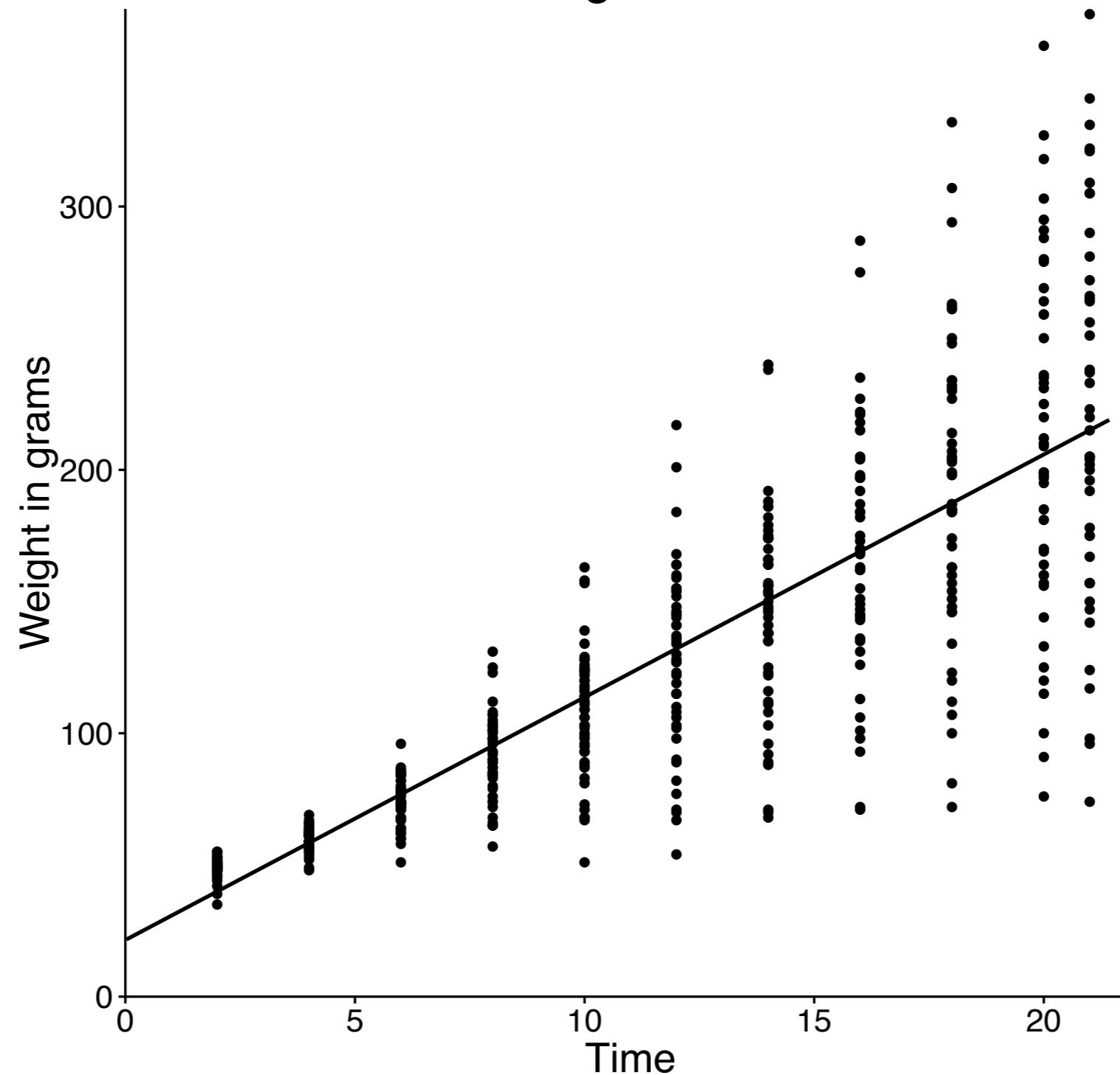
a = intercept

b = slope

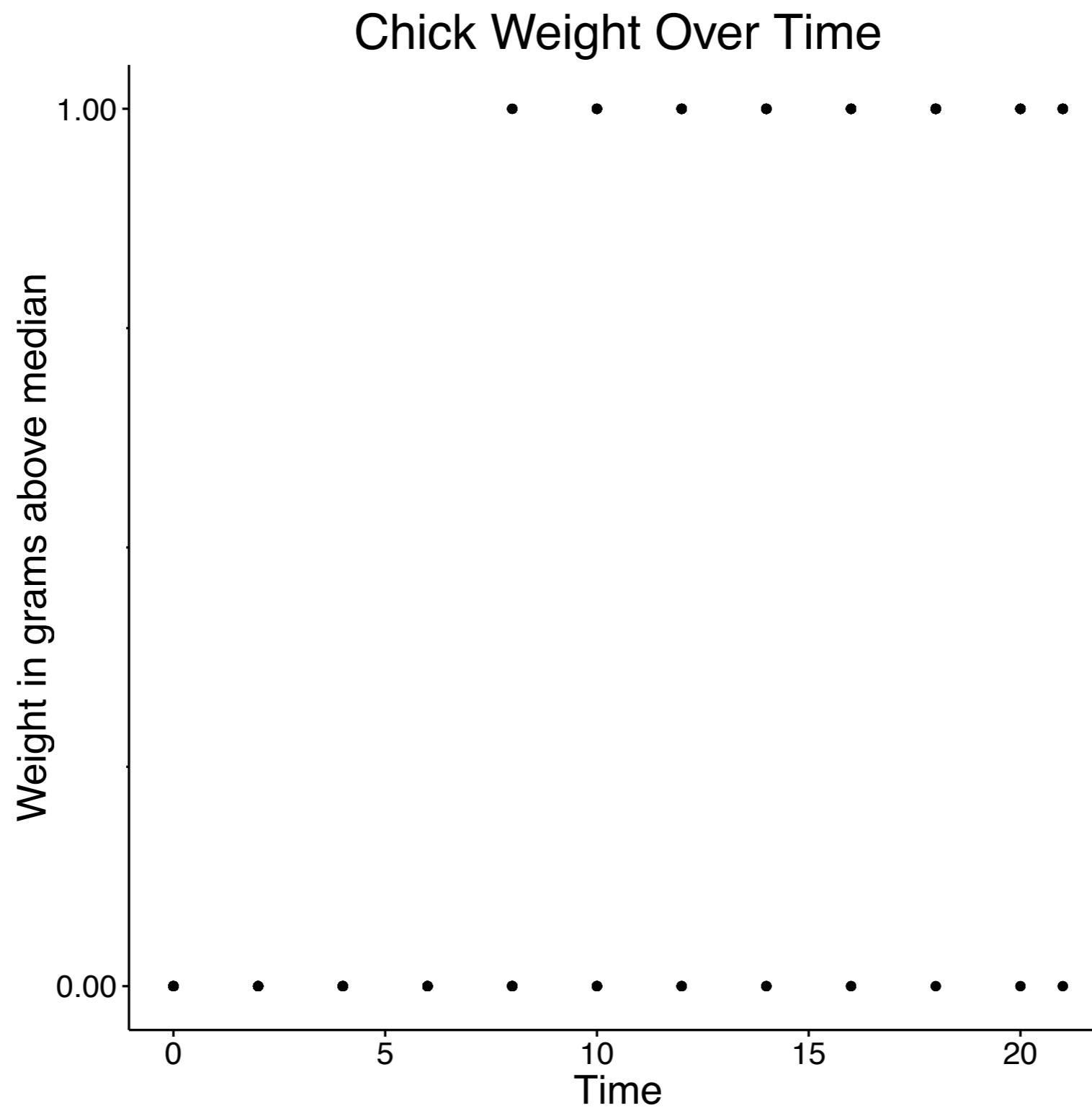
x_i = specific x-values (independent variable)

$$y_i = a + b x_i + e_i$$

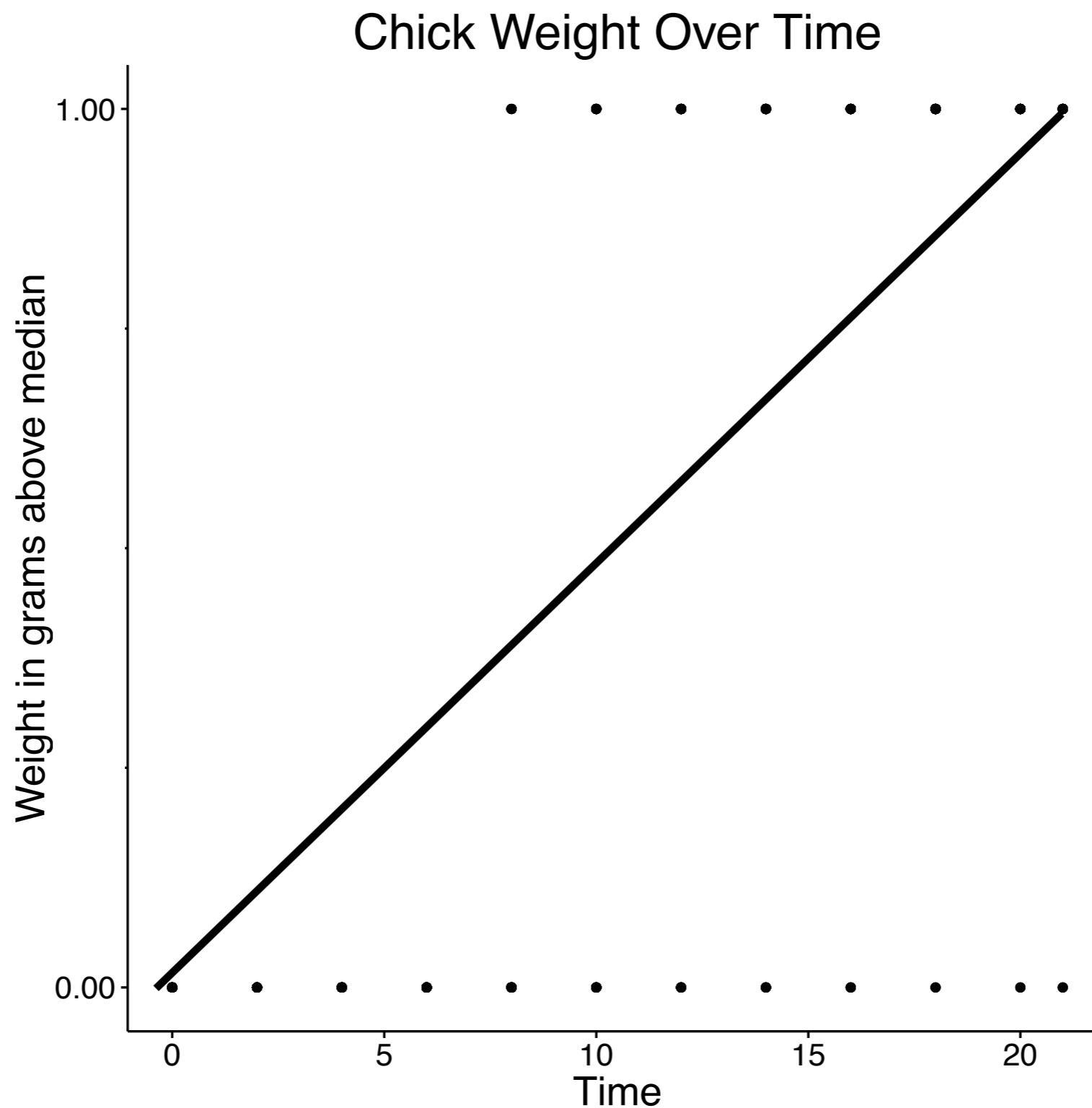
Chick Weight Over Time



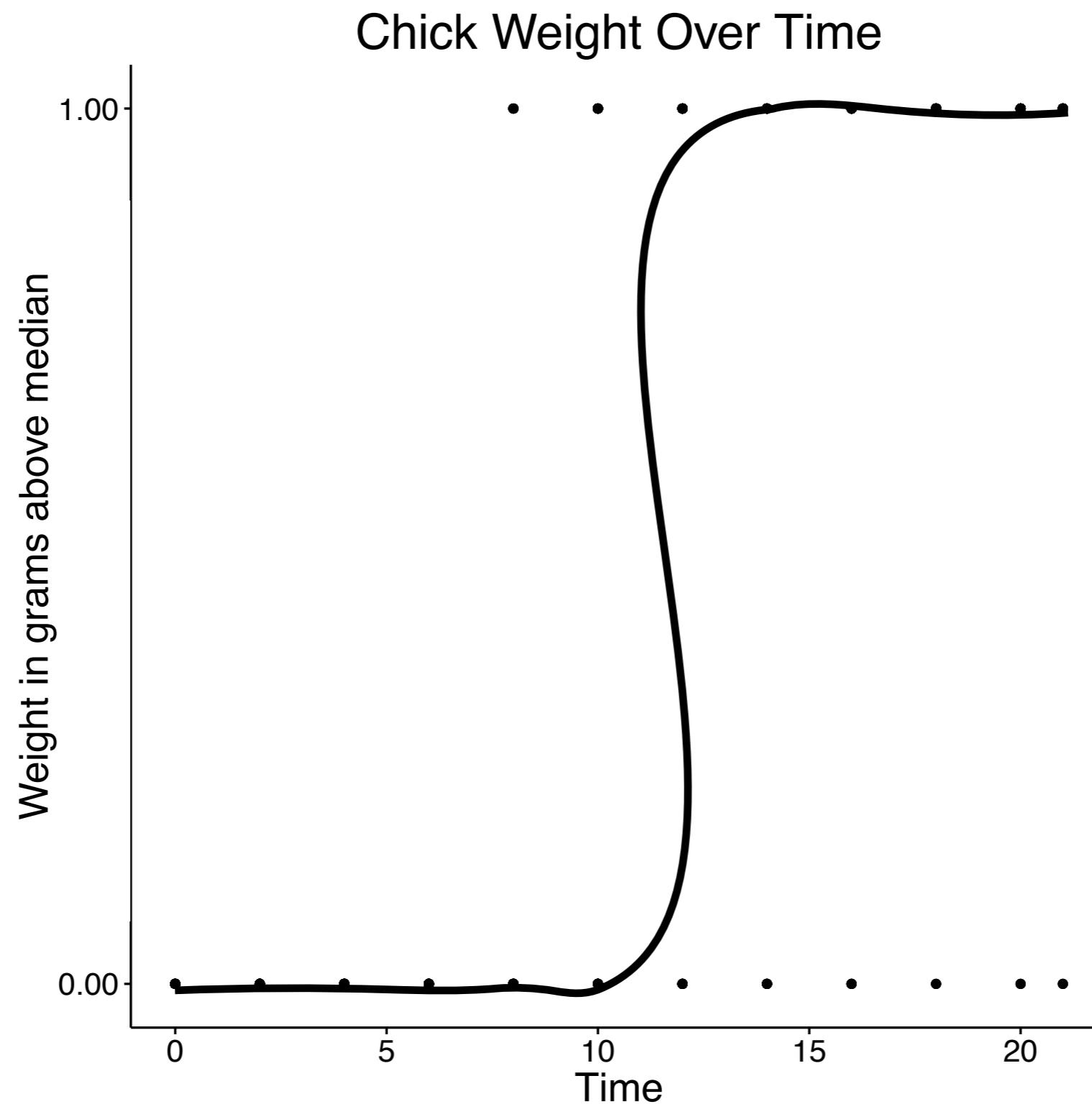
$$\log[p/(1-p)]_i = a + b x_i$$



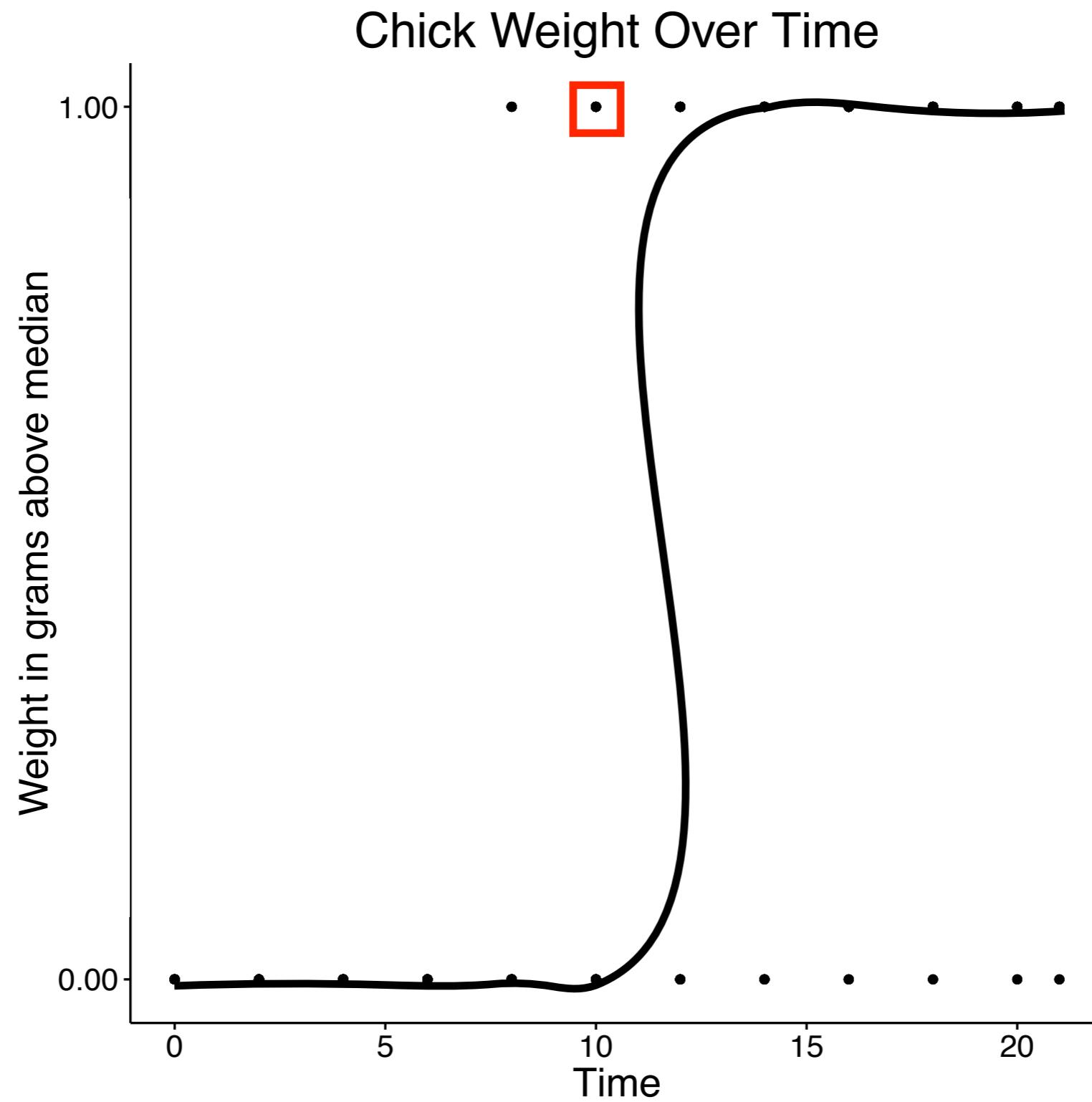
$$\log[p/(1-p)]_i = a + b x_i$$



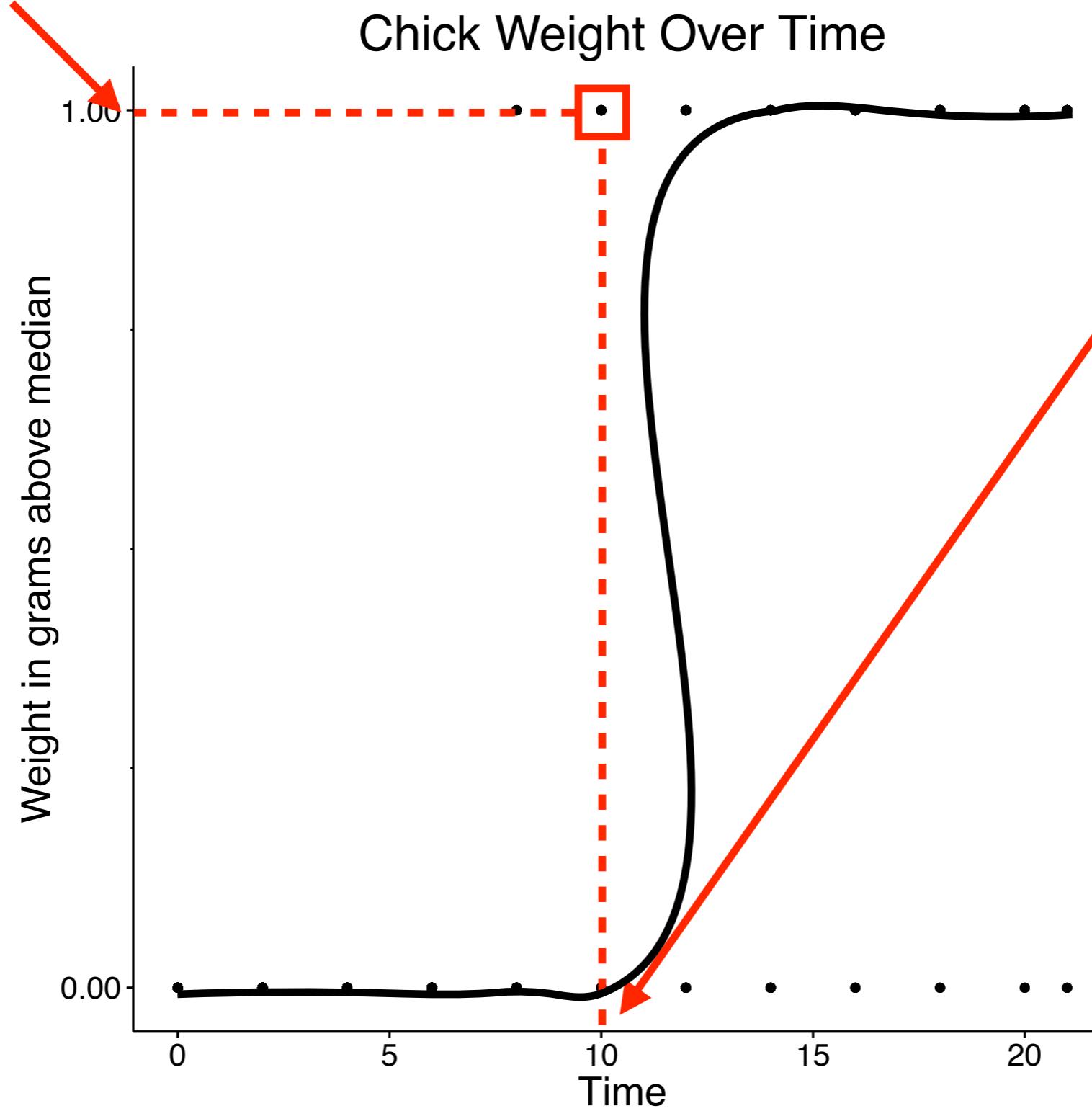
$$\log[p/(1-p)]_i = a + bX_i$$



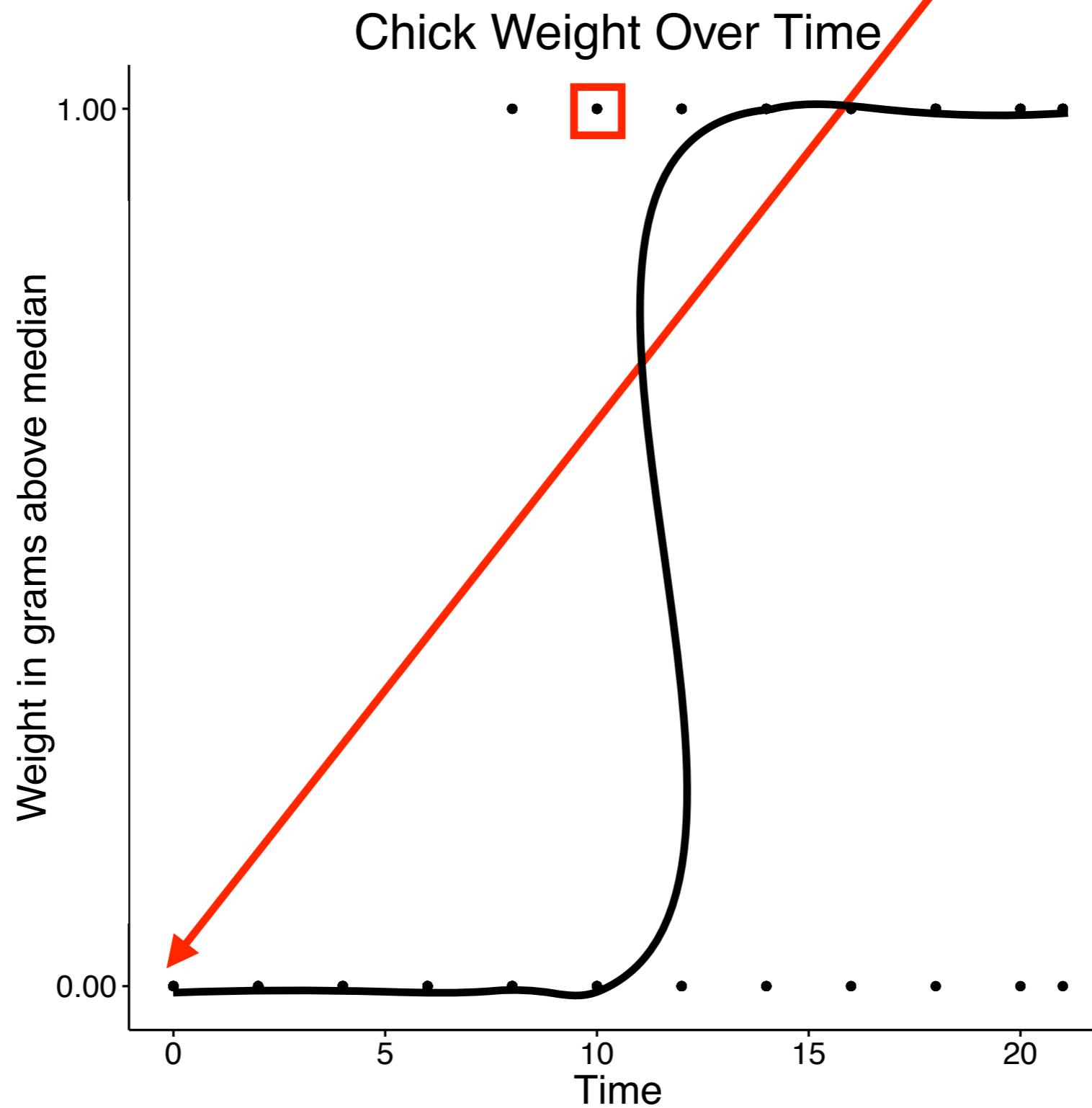
$$\log[p/(1-p)]_i = a + bX_i$$



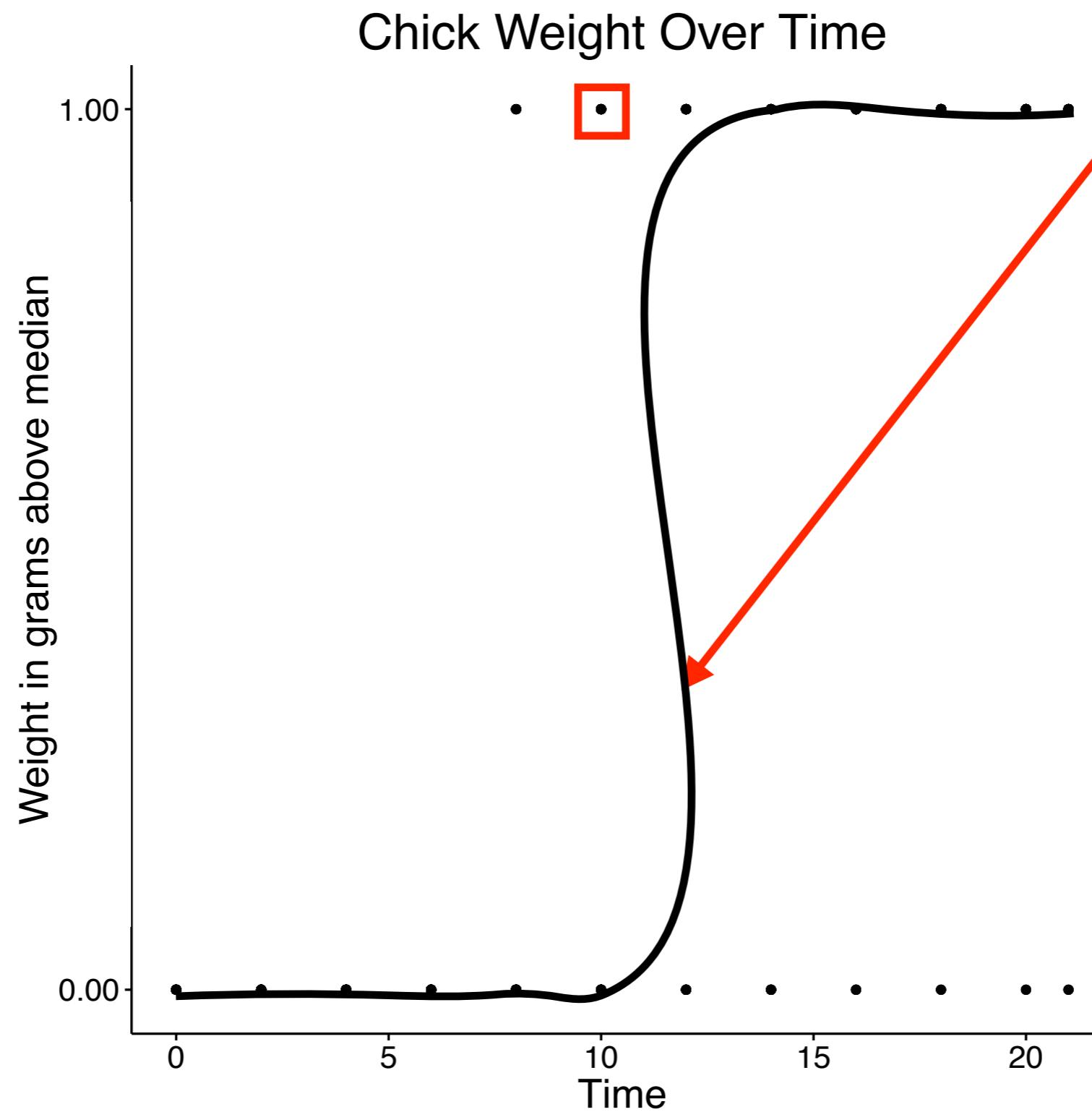
$$\log[p/(1-p)]_i = a + b x_i$$



$$\log[p/(1-p)]_i = a + b x_i$$

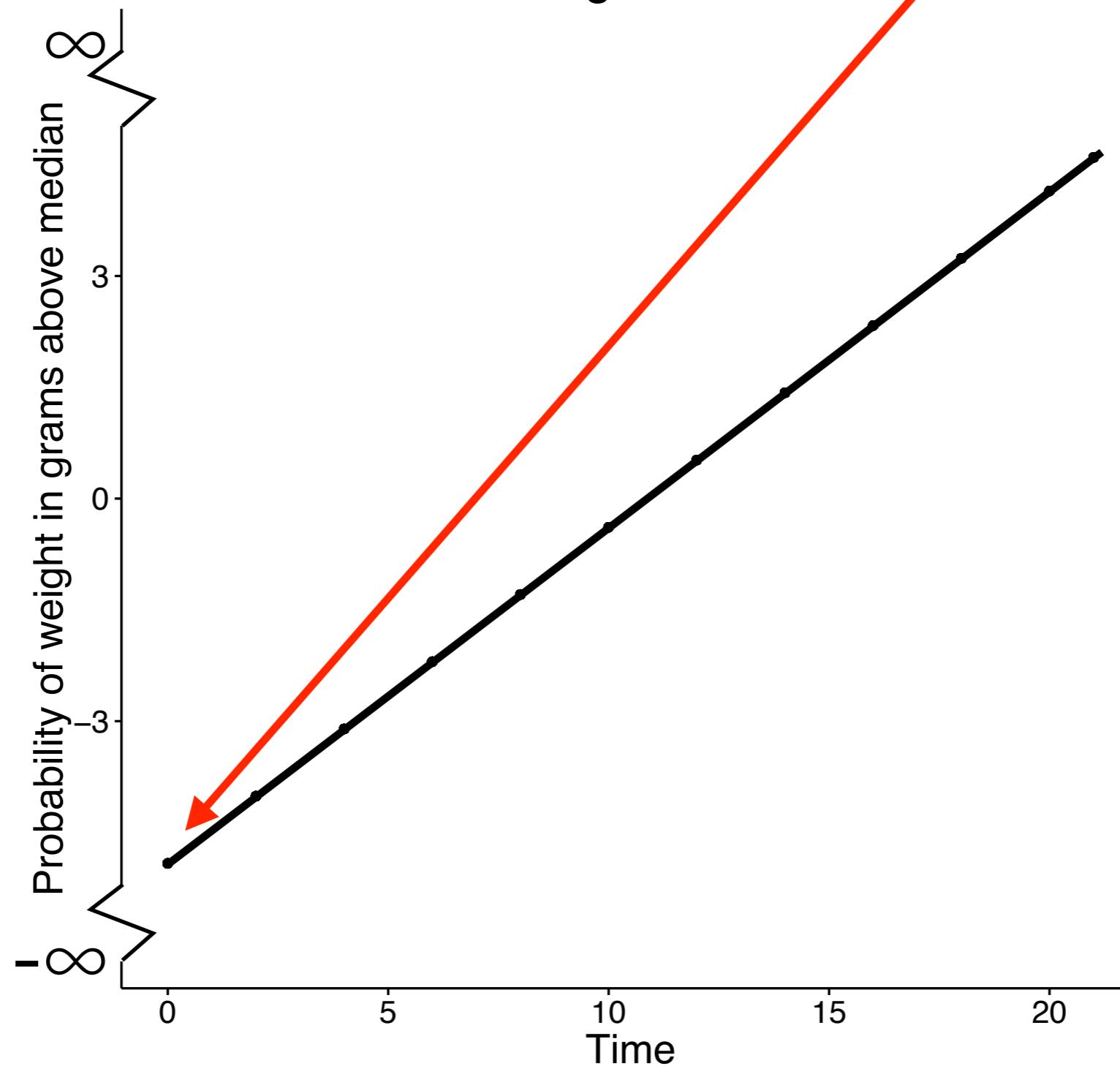


$$\log[p/(1-p)]_i = a + b x_i$$



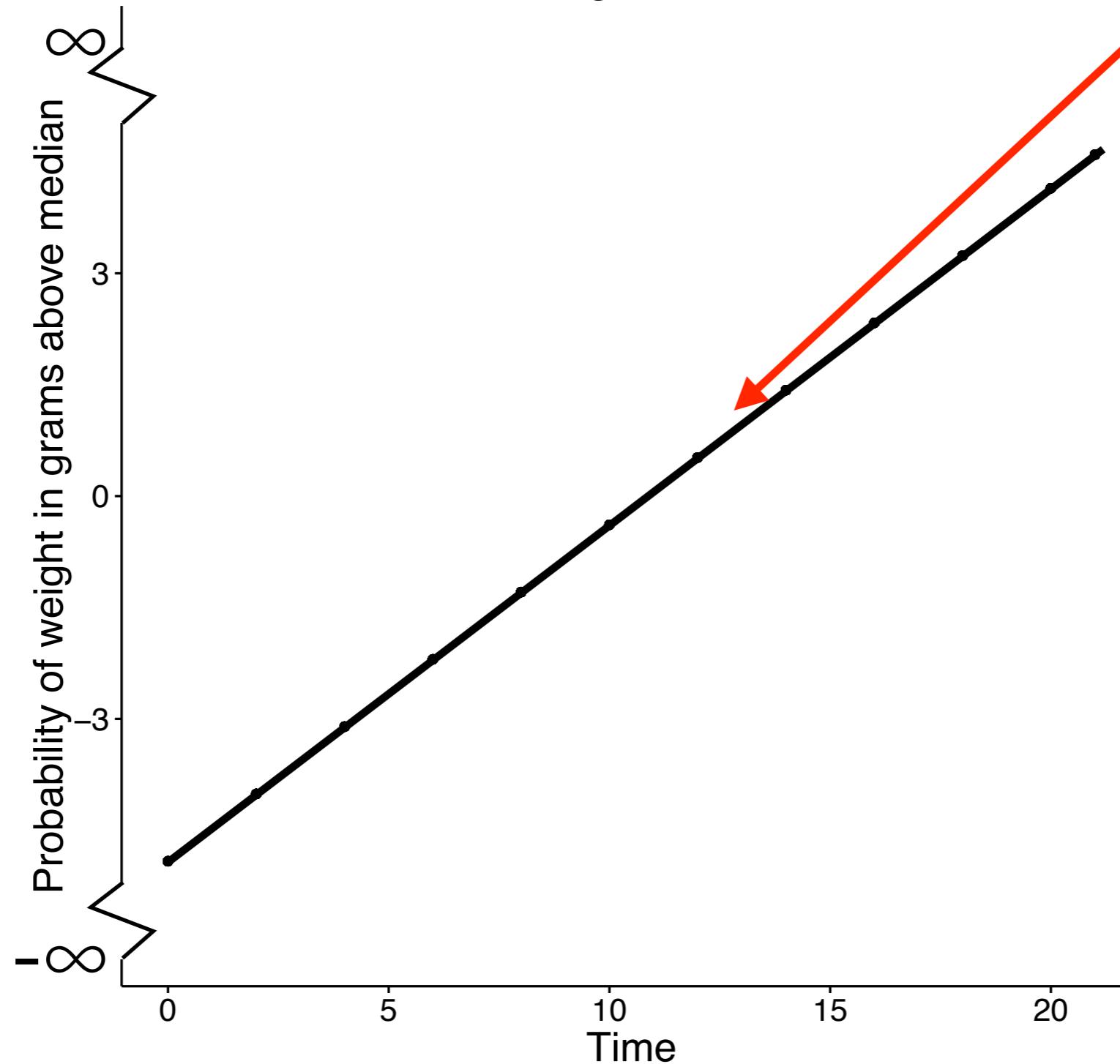
$$\log[p/(1-p)]_i = a + b x_i$$

Chick Weight Over Time

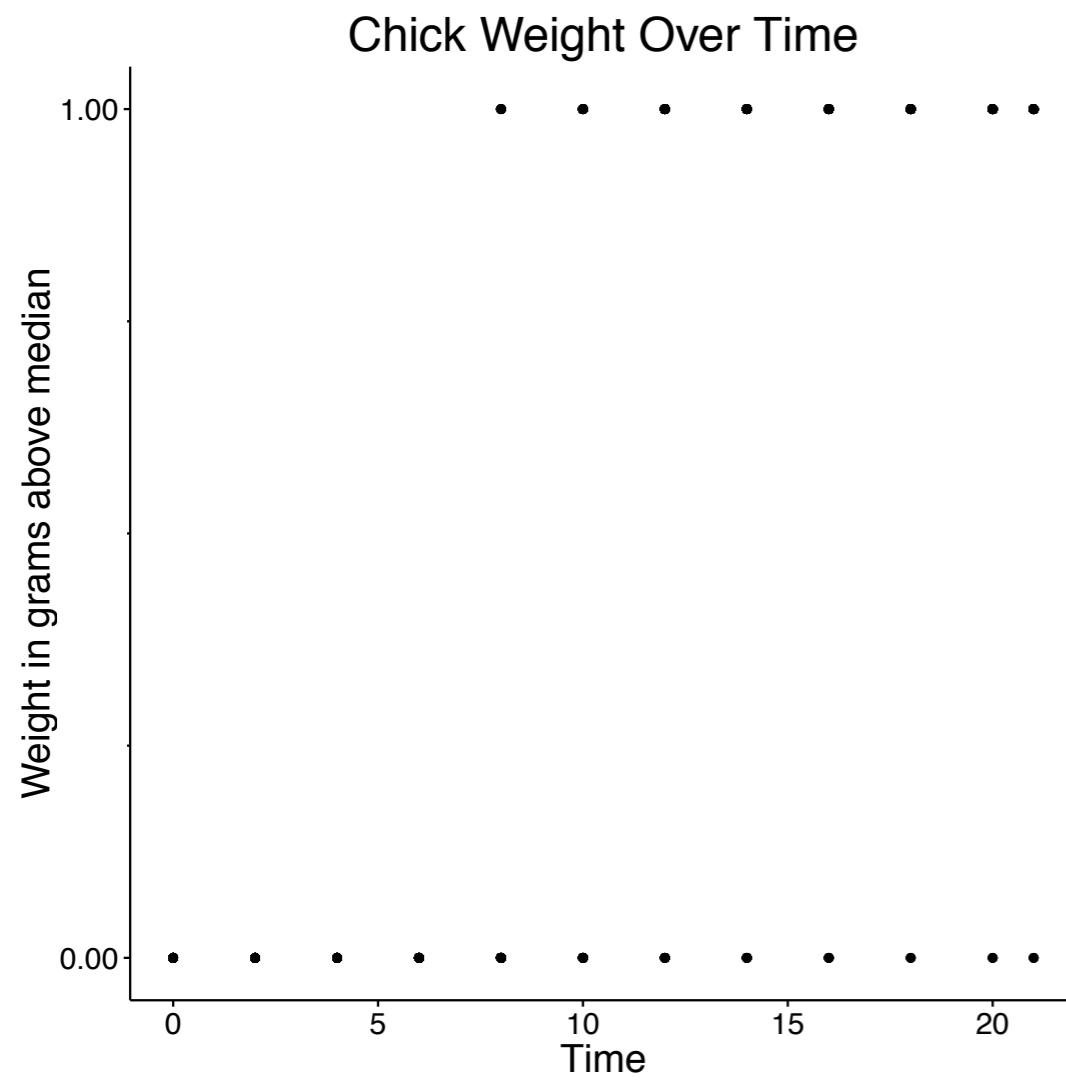


$$\log[p/(1-p)]_i = a + b x_i$$

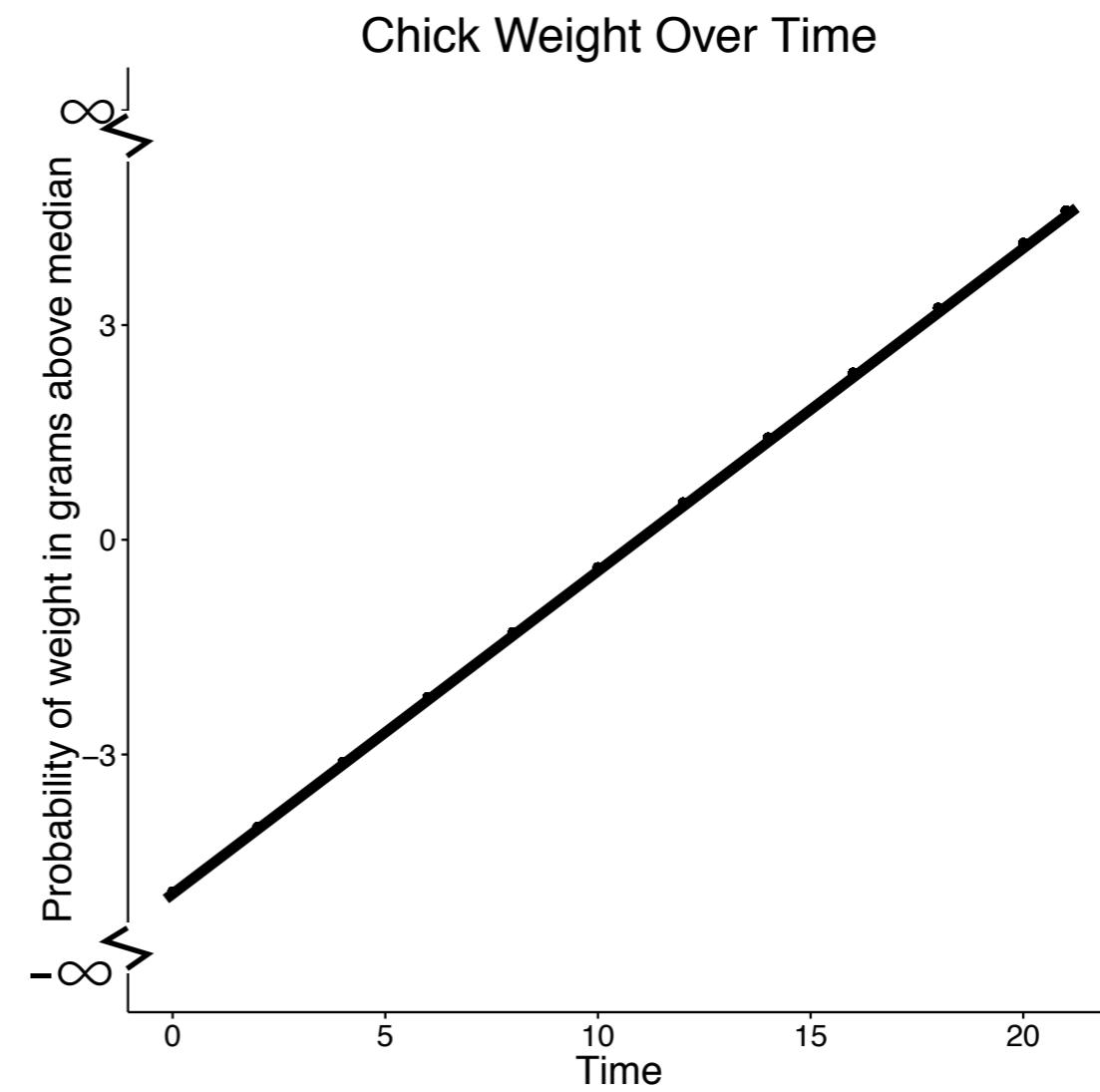
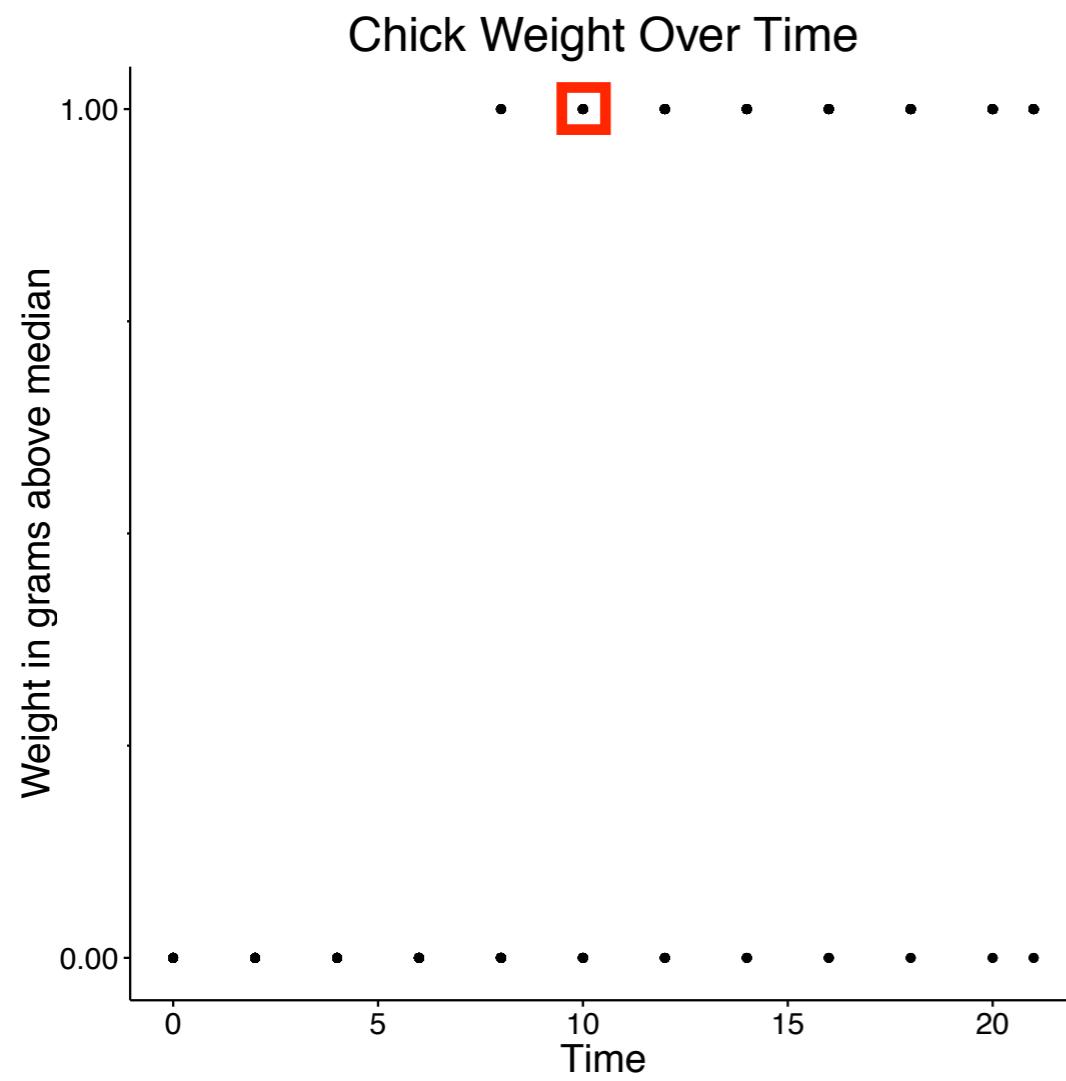
Chick Weight Over Time



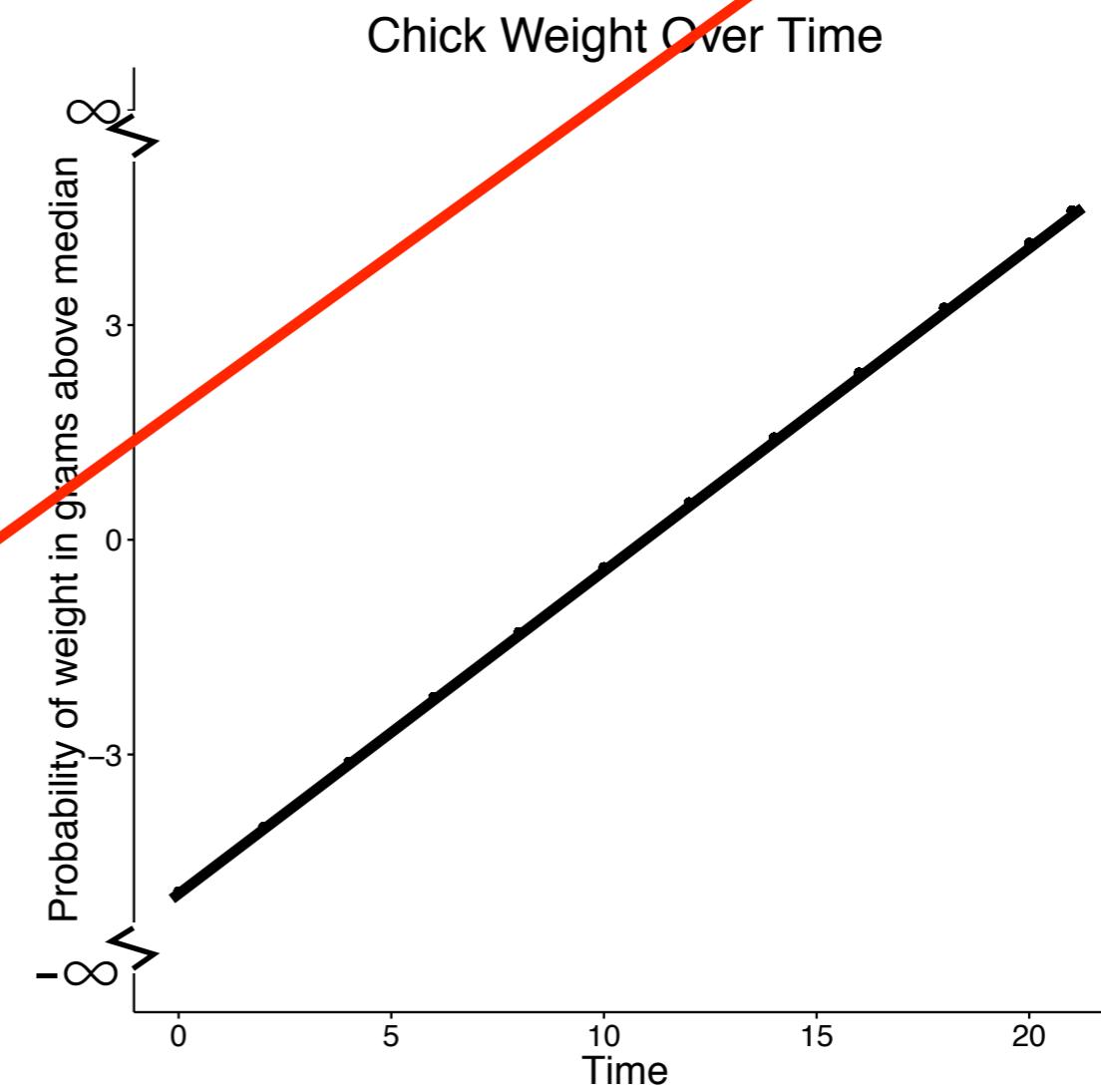
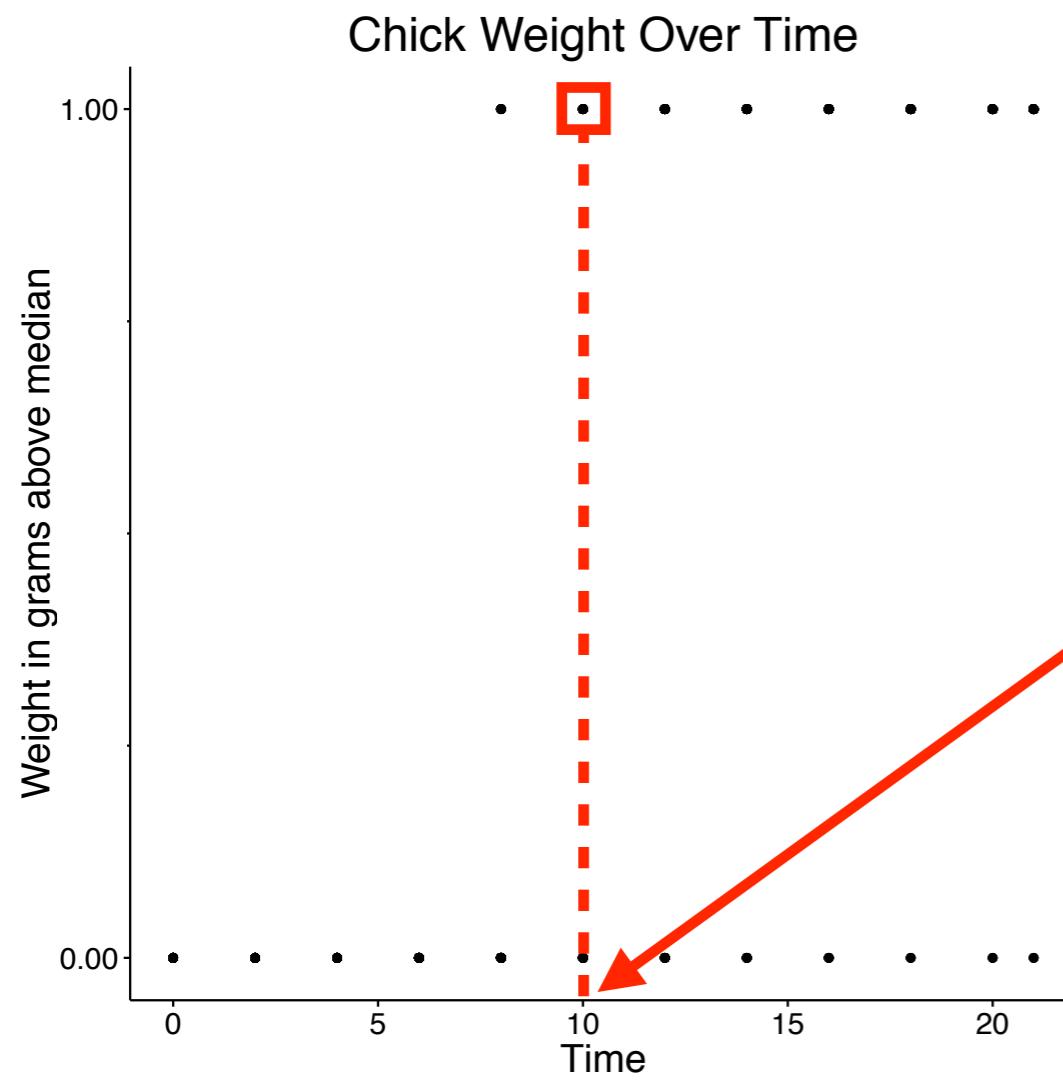
$$\log[p/(1-p)]_i = a + b x_i$$



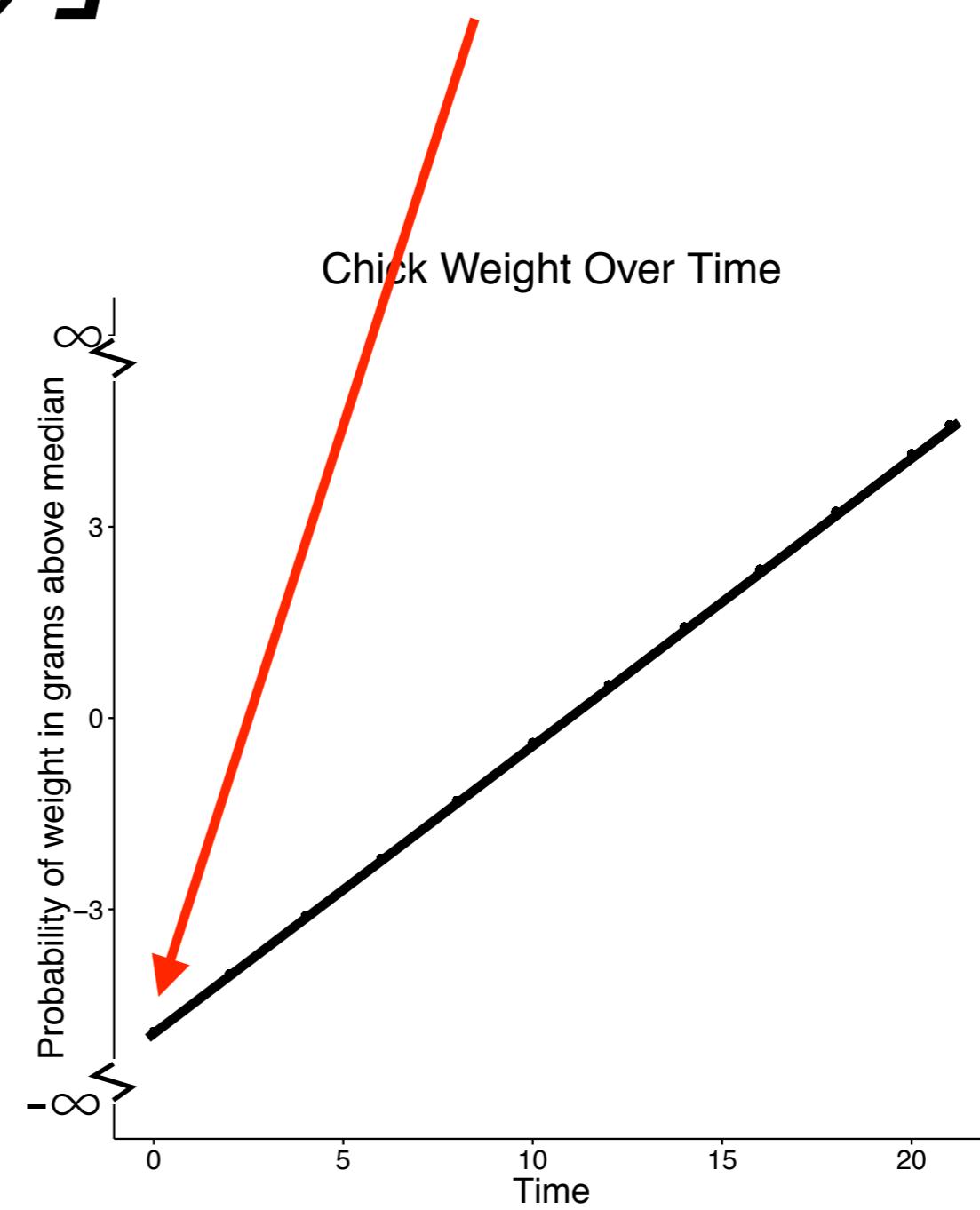
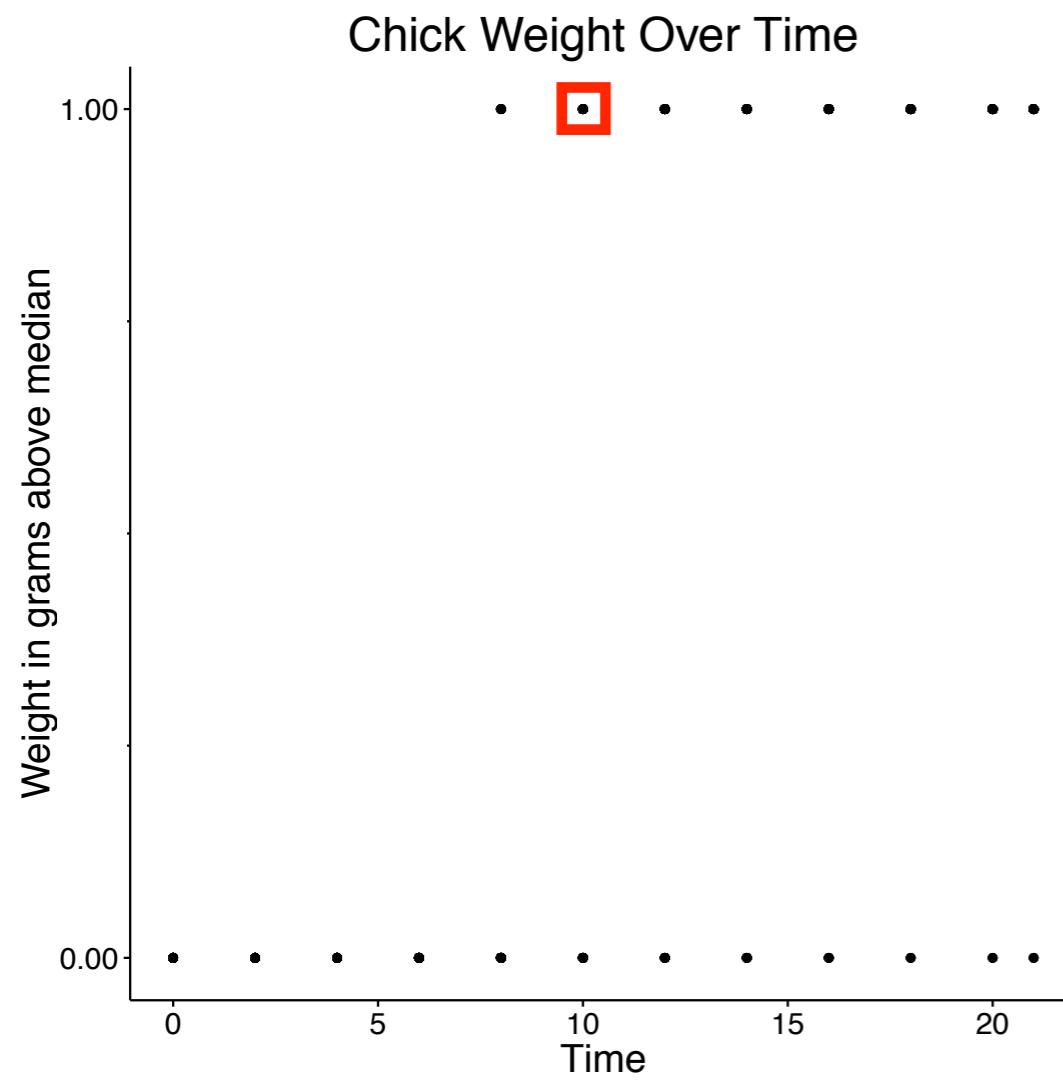
$$\log[p/(1-p)]_i = a + b x_i$$



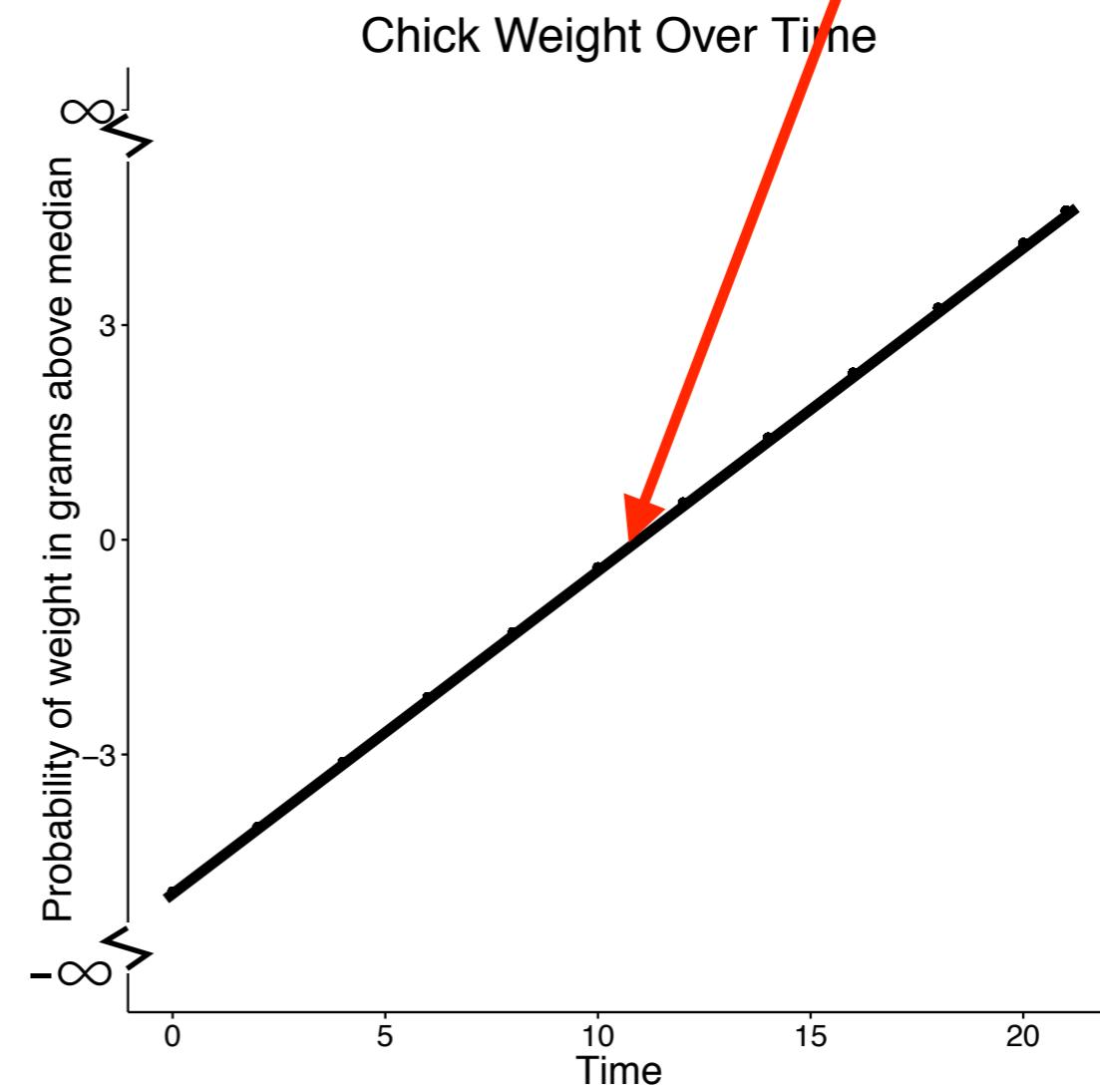
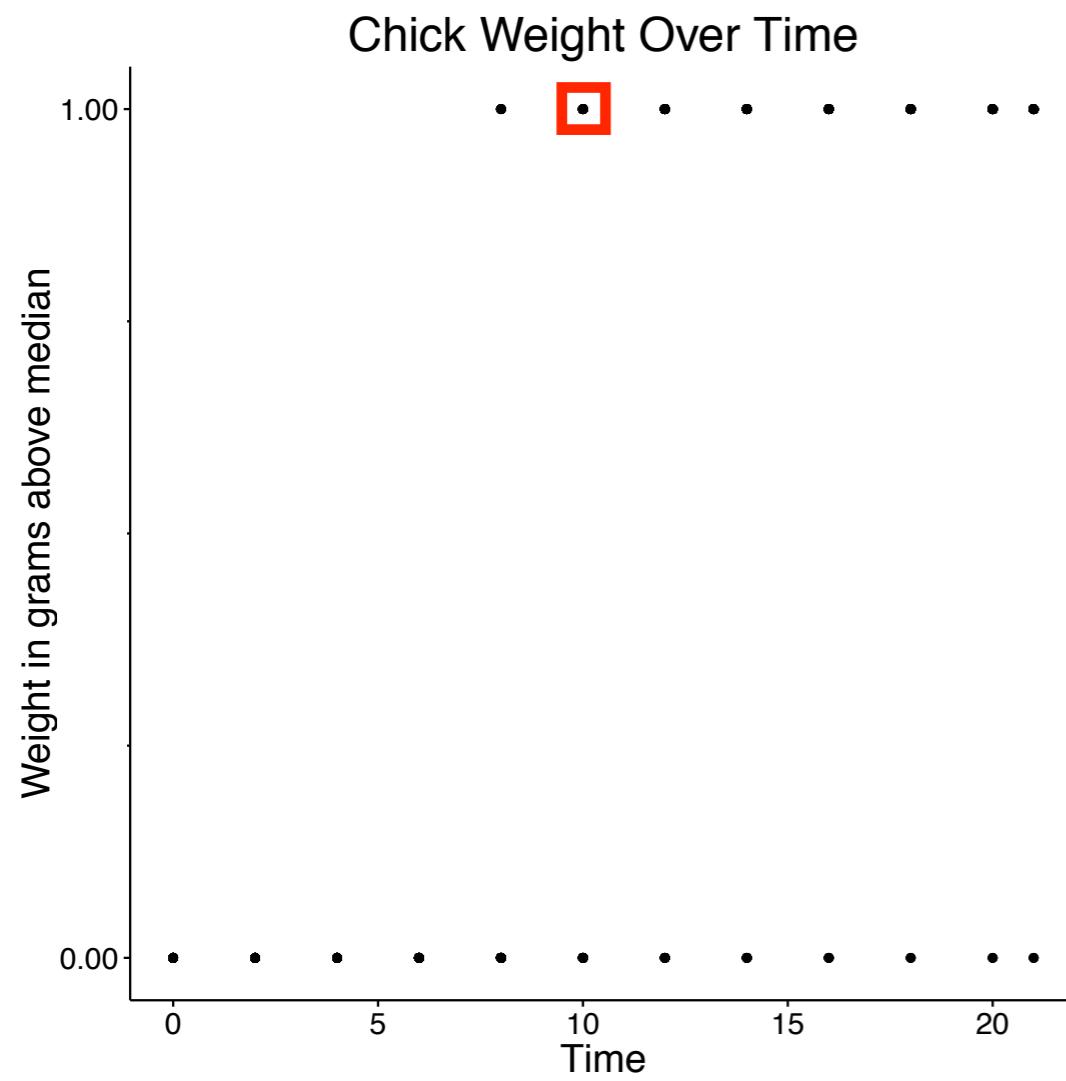
$$\log[p/(1-p)]_i = a + b x_i$$



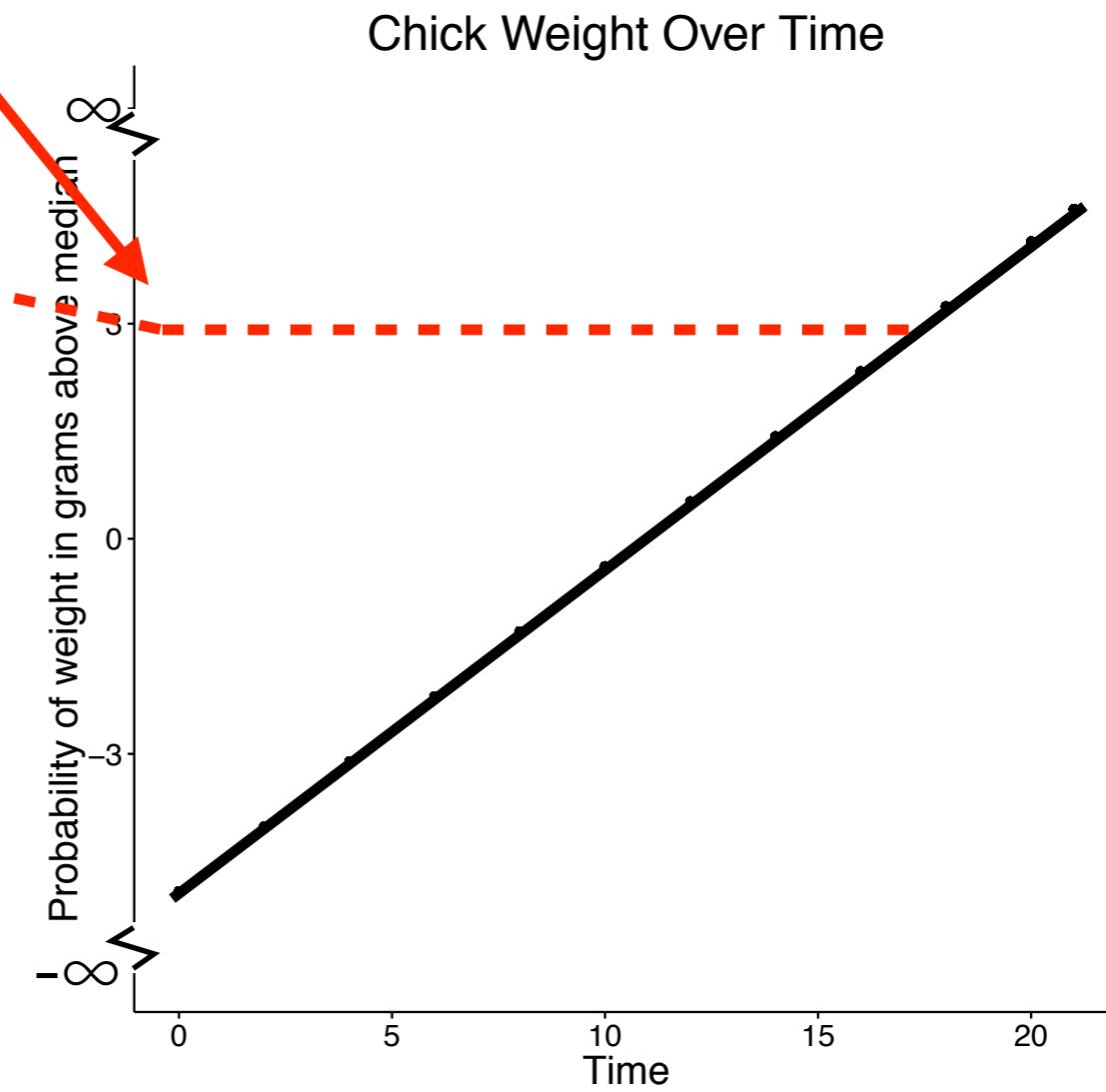
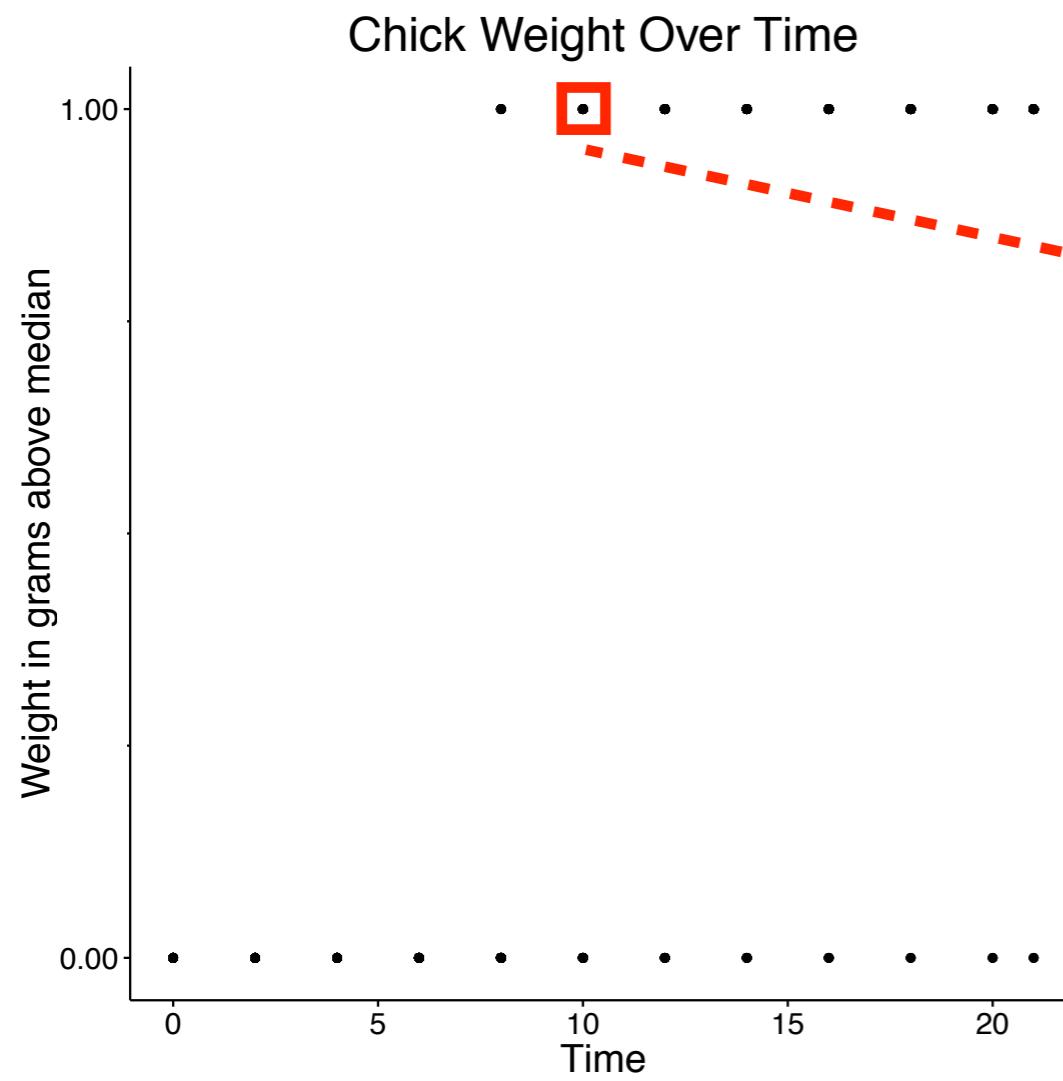
$$\log[p/(1-p)]_i = a + bx_i$$



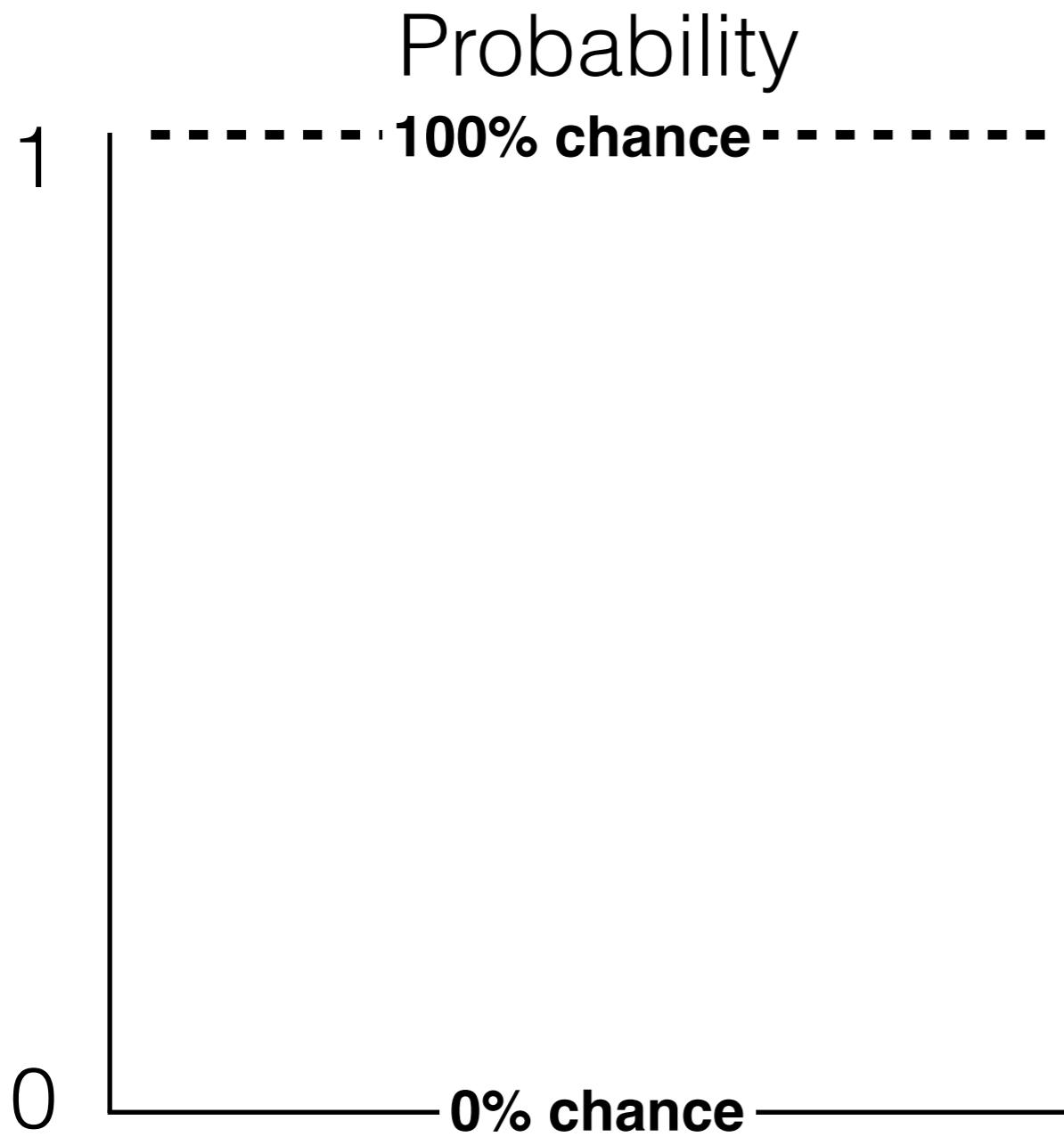
$$\log[p/(1-p)]_i = a + bx_i$$



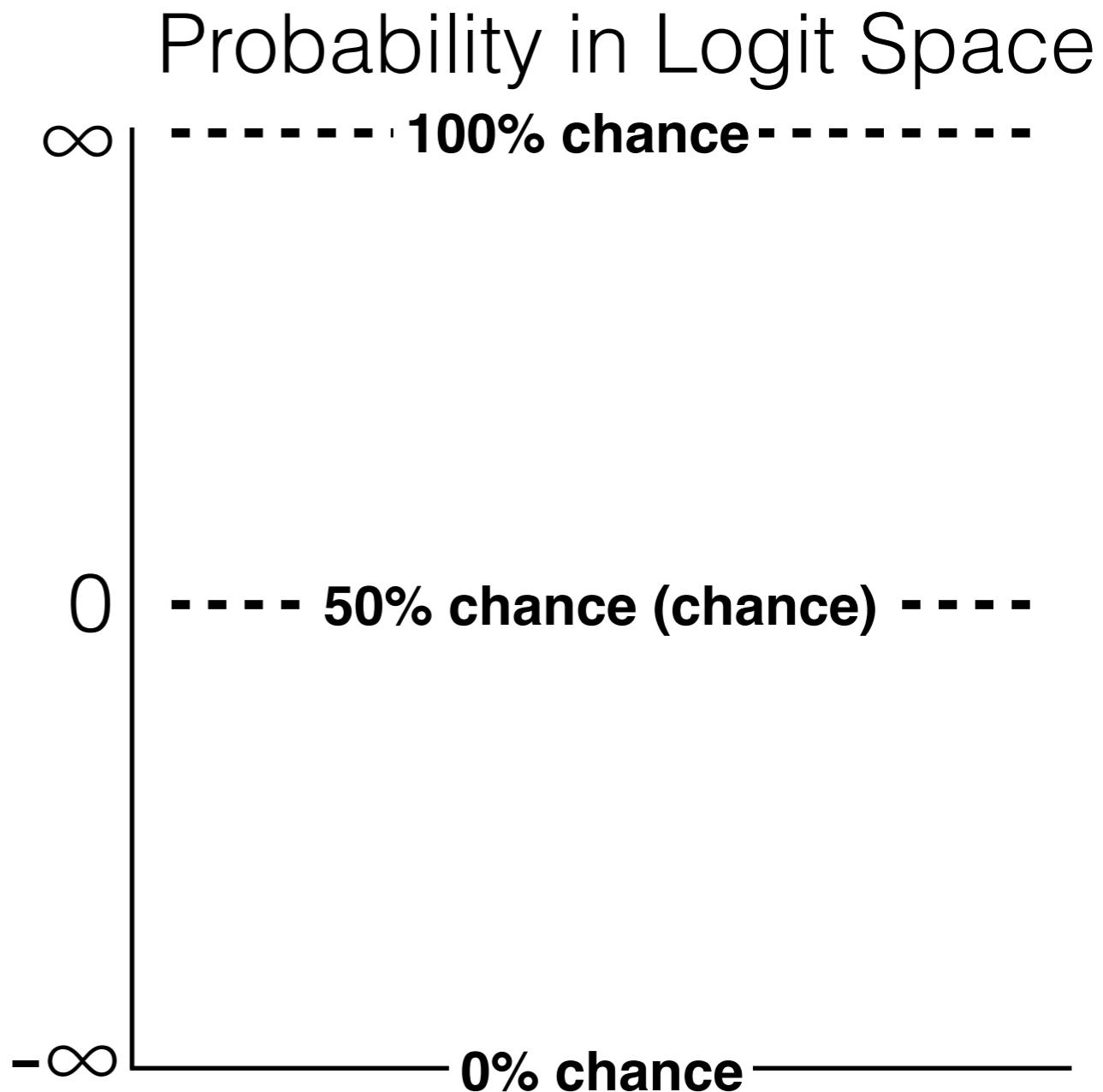
$$\log[p/(1-p)]_i = a + b x_i$$



INPUT



MODEL COEFFICIENTS



R Code

$$\log[p/(1-p)]_i = a + b x_i$$

```
glm(weight_above_median ~ Time,  
family="binomial")
```

Call:
glm(formula = weight_median_above ~ Time, family = "binomial",
data = chickweight_lesson)

Deviance Residuals:
Min 1Q Median 3Q Max
-3.0360 -0.2962 -0.1208 0.4303 1.7519

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.9167 0.41438 -11.87 <2e-16 ***
Time 0.45311 0.03611 12.55 <2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 801.22 on 577 degrees of freedom
Residual deviance: 347.16 on 576 degrees of freedom
AIC: 351.16

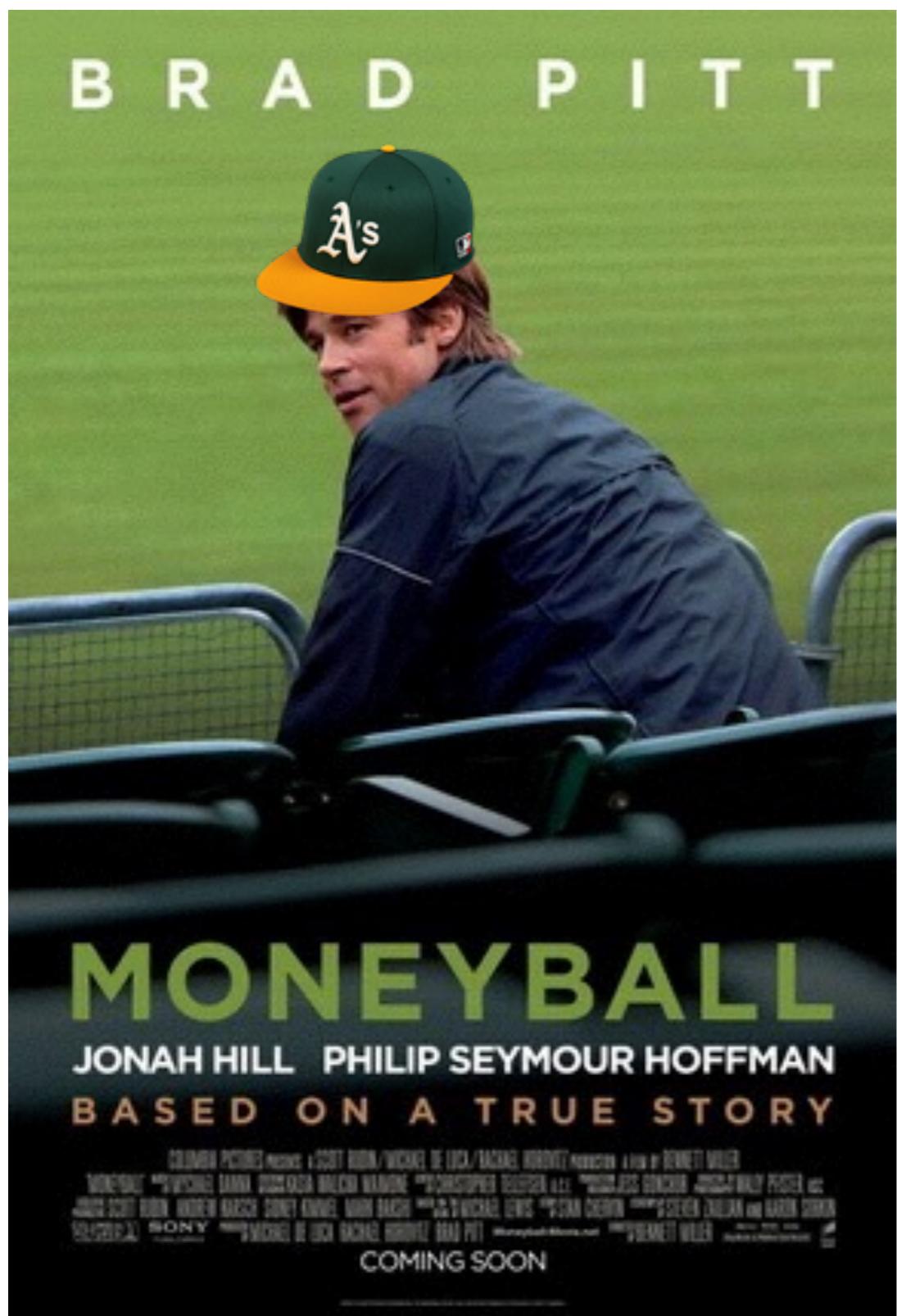
Number of Fisher Scoring iterations: 6

Lab

Data set: The San Francisco Giants 2010 Baseball Season

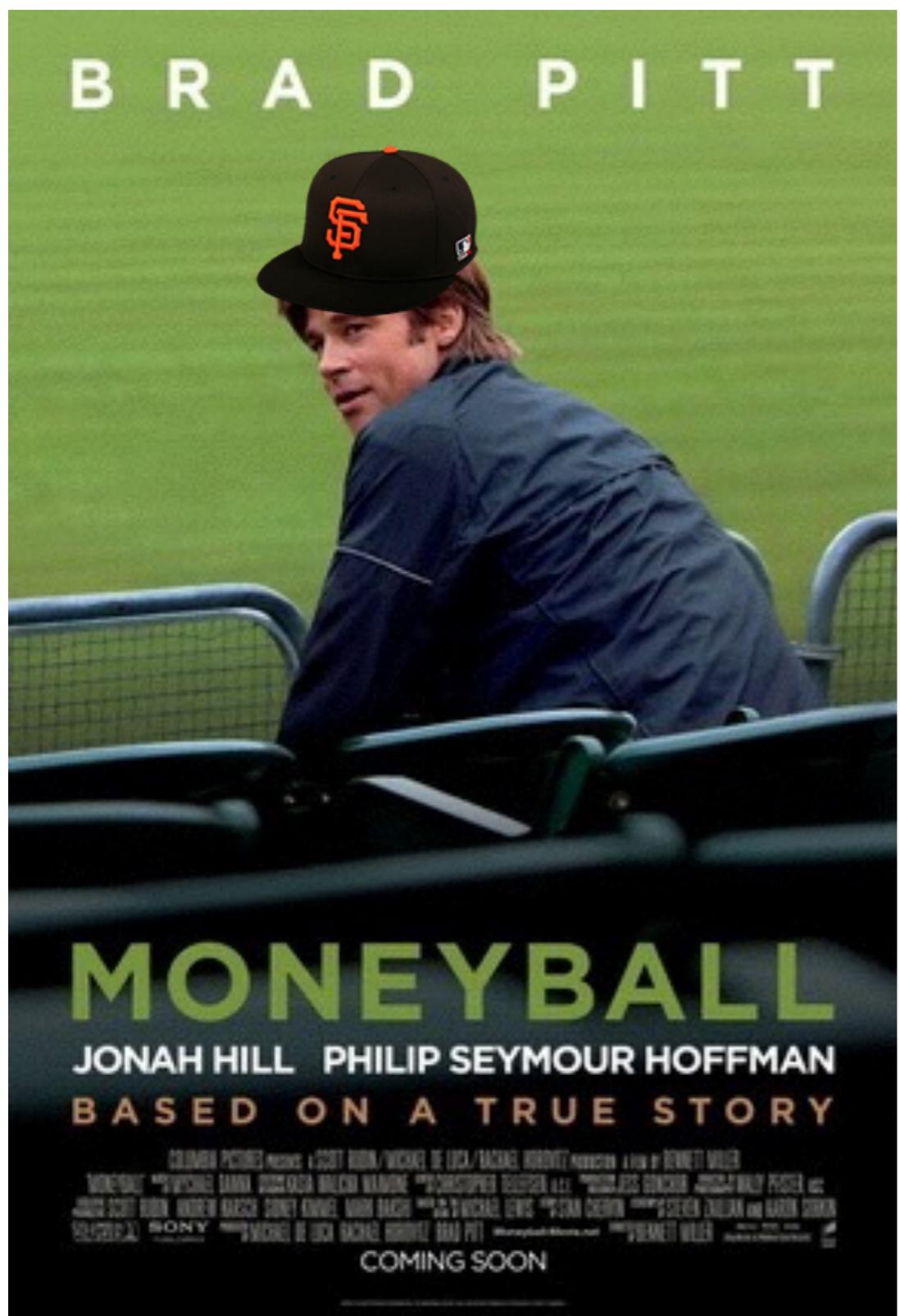
source: retrosheet.org

B R A D P I T T



The logo for QR, featuring a stylized lowercase 'q' and a large, bold, blue uppercase 'R' inside a thick, dark grey oval.

B R A D P I T T



The image shows the logo for the Quality Rating (QR) system. It consists of a large, stylized, three-dimensional letter 'R' in a dark blue-grey color. The 'R' is set against a background that features a large, thin, dark grey oval shape that overlaps the top and bottom of the 'R'. The 'R' itself has a thick, rounded, and slightly recessed center, giving it a three-dimensional appearance. The overall design is clean and modern.

Data set: The San Francisco Giants 2010 Baseball Season

Full Season: Did the Giants win more games before or after the All-Star break?

Buster Posey: Are the Giant's more likely to win in games where Buster Posey was walked at least once?

Full Season

logit p_i = win or loss
a = ? - from model
b = ? - from model
 x_i = All Star break

Buster Posey

logit p_i = win or loss
a = ? - from model
b = ? - from model
 x_i = walked

dplyr

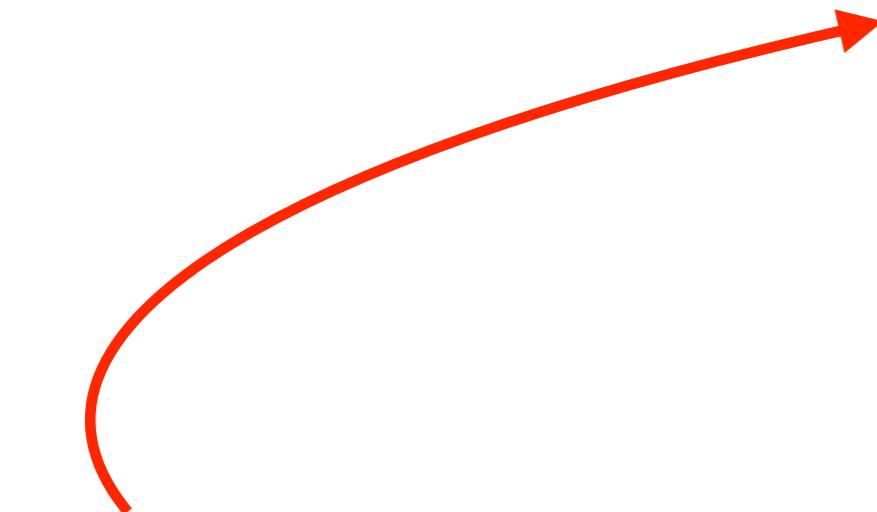
```
data_clean = data
```

dplyr

```
data_clean = data %>%
```

dplyr

```
data_clean = data %>%  
  mutate(
```



)

dplyr

```
data_clean = data %>%  
  mutate(home_visitor =
```

verb

new
variable



)

dplyr

```
data_clean = data %>%  
  mutate(home_visitor =
```

The diagram illustrates a `dplyr` mutate call with the following annotations:

- verb**: Points to the word `mutate`.
- new variable**: Points to the variable name `home_visitor`.
- conditional statement**: Points to the `ifelse` function.

```
mutate(home_visitor =  
  ifelse(  
    ))
```

dplyr

```
data_clean = data %>%  
  mutate(home_visitor =  
    ifelse(home_team
```

verb

new
variable

conditional
statement

variable

dplyr

```
data_clean = data %>%  
  mutate(home_visitor =
```

verb

new
variable

conditional
statement

variable

relationship
marker

```
  ifelse(home_team ==
```

```
  ))
```

dplyr

```
data_clean = data %>%  
  mutate(home_visitor =  
    ifelse(home_team == "SFN"  
    ))
```

verb

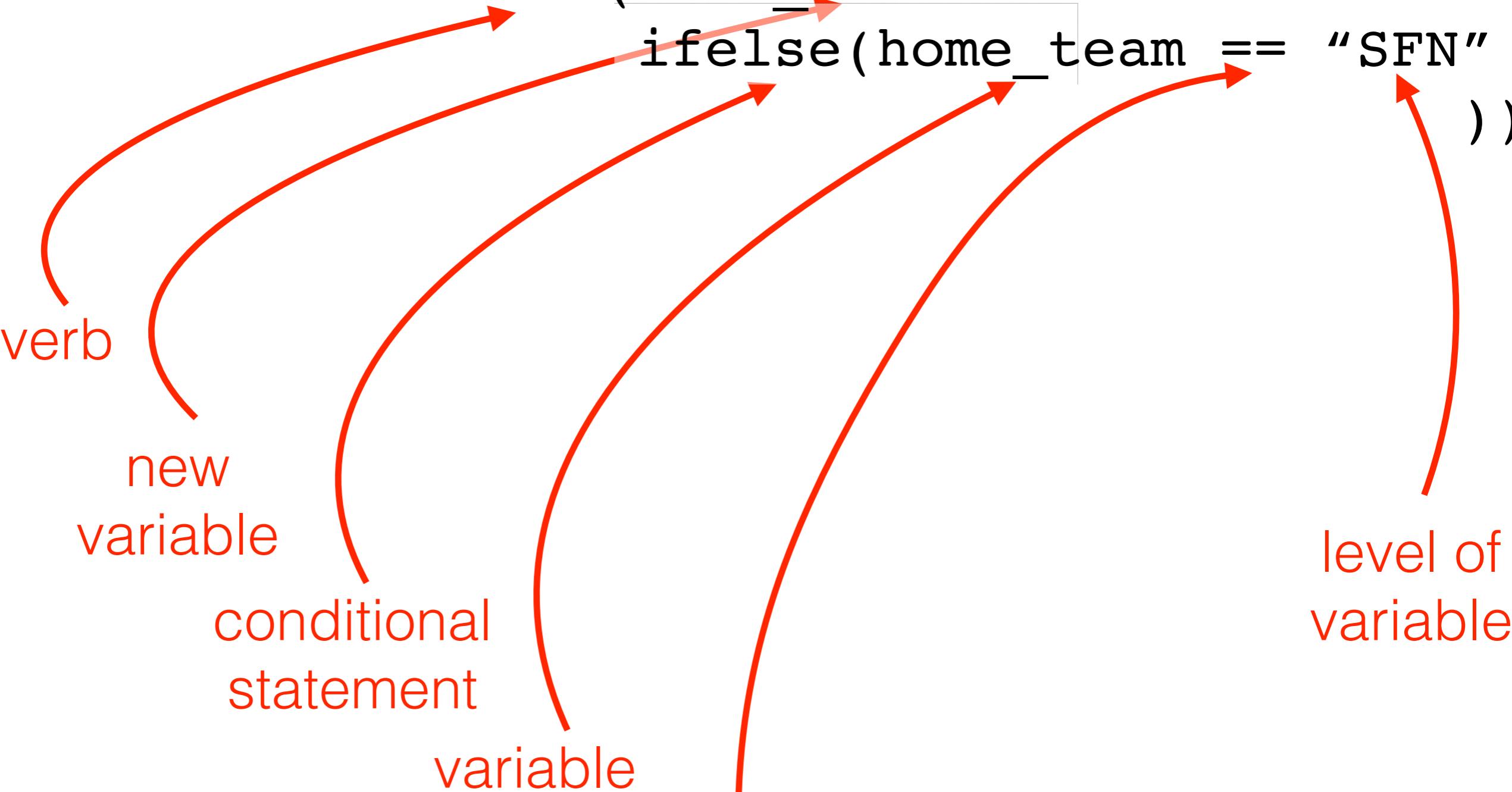
new variable

conditional statement

variable

relationship marker

level of variable



dplyr

```
data_clean = data %>%  
  mutate(home_visitor =  
    ifelse(home_team == "SFN",  
          "home",  
          ))
```

The diagram illustrates the structure of the `mutate` statement with red annotations:

- verb**: Points to the word `mutate`.
- new variable**: Points to the variable `home_visitor`.
- conditional statement**: Points to the `ifelse` function.
- variable**: Points to the variable `home_team`.
- relationship marker**: Points to the `==` operator.
- level of new variable if true**: Points to the value `"home"`.
- level of variable**: Points to the value `"SFN"`.

dplyr

```
data_clean = data %>%
```

```
  mutate(home_visitor =
```

```
    ifelse(home_team == "SFN",  
           "home", "visitor"))
```

verb

new
variable

conditional
statement

variable

relationship
marker

level of
new
variable
if true

level of new
variable
if false

level of
variable

dplyr

data_posey_clean = data_posey

dplyr

```
data_posey_clean = data_posey %>%
```

dplyr

```
data_posey_clean = data_posey %>%  
  inner_join( )
```



two table
verb

dplyr

```
data_posey_clean = data_posey %>%  
  inner_join(data_clean)
```



two table
verb

data frame

dplyr

```
data_posey_clean = data_posey %>%  
  inner_join(data_clean)
```

two table
verb

data frame

```
data_posey + data_clean = data_posey_clean
```

date	opponent
20100529	ARI
20100530	ARI
20100531	COL
20100601	COL

date	day_of_week
20100405	Mon
20100406	Tue
20100529	Sat
20100530	Sun

date	opponent	day_of_week
20100529	ARI	Sat
20100530	ARI	Sun

dplyr

```
data_posey_clean = data_posey %>%  
  inner_join(data_clean)
```

two table
verb



data frame

```
data_posey + data_clean = data_posey_clean
```

date	opponent
20100529	ARI
20100530	ARI

date	opponent
20100531	COL

date	opponent
20100601	COL

date	day_of_week
20100405	Mon
20100406	Tue

date	day_of_week
20100529	Sat

date	day_of_week
20100530	Sun

date	opponent	day_of_week
20100529	ARI	Sat
20100530	ARI	Sun

date	opponent	day_of_week
20100531	COL	Sat

dplyr

```
data_posey_clean = data_posey %>%  
  inner_join(data_clean)
```

two table
verb

data frame

```
data_posey + data_clean = data_posey_clean
```

date	at_bats
20100529	4
20100530	5
20100531	3
20100601	4

date	day_of_week
20100405	Mon
20100406	Tue
20100529	Sat
20100530	Sun

date	opponent	day_of_week
20100529	ARI	Sat
20100530	ARI	Sun

dplyr

data_figs_sum = data_figs

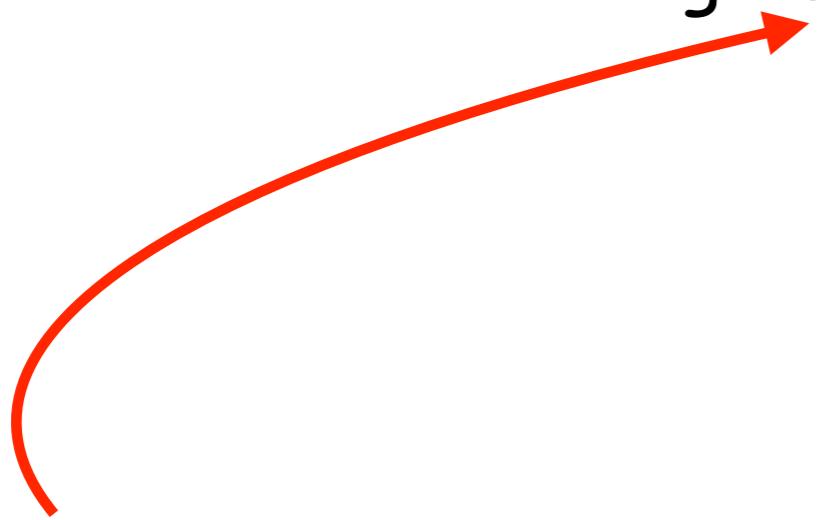
dplyr

```
data_figs_sum = data_figs %>%
```

dplyr

```
data_figs_sum = data_figs %>%  
  group_by( )
```

verb



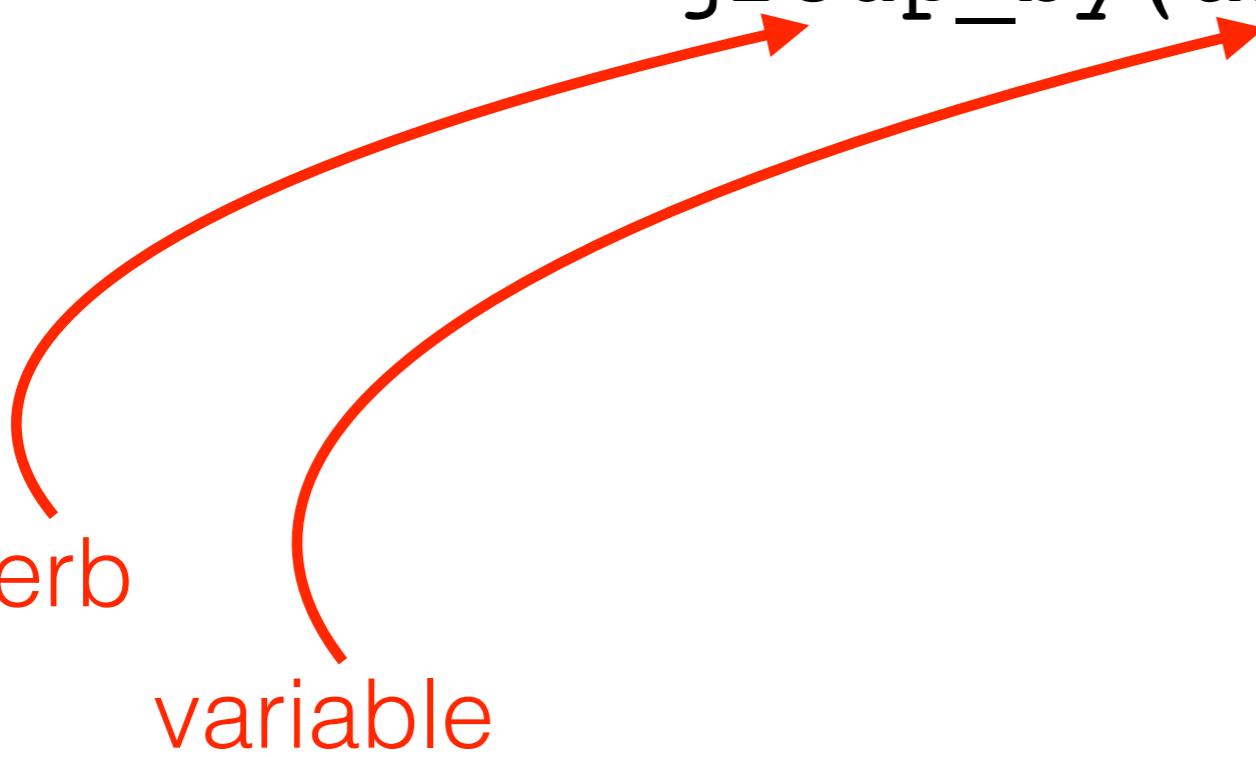
dplyr

```
data_figs_sum = data_figs %>%  
  group_by(allstar_break)
```

verb

variable

to group by



dplyr

```
data_figs_sum = data_figs %>%  
  group_by(allstar_break) %>%  
  summarise(  
  )
```

verb

variable

to group by

verb

dplyr

```
data_figs_sum = data_figs %>%  
  group_by(allstar_break) %>%  
  summarise(wins_perc  
)
```

verb

variable

to group by

verb

new
variable

dplyr

```
data_figs_sum = data_figs %>%  
  group_by(allstar_break) %>%  
  summarise(wins_perc =  
            mean(win) * 100)
```

verb

variable

to group by

verb

new
variable

function

to summarize by

dplyr

```
data_figs_sum = data_figs %>%  
  group_by(allstar_break) %>%  
  summarise(wins_perc =  
            mean(win) * 100) %>%  
  ungroup()
```

The diagram illustrates the flow of data and operations in a dplyr pipe. It starts with the assignment of `data_figs` to `data_figs_sum`. The pipe then enters the first `%>%` block, where the `group_by` function is applied to the `allstar_break` variable. The pipe then enters the second `%>%` block, where the `summarise` function is applied. Inside the `summarise` block, the `mean` function is used to calculate the mean of the `win` variable, and this result is multiplied by 100 to get the `wins_perc` variable. Finally, the pipe exits the `summarise` block and enters the third `%>%` block, where the `ungroup` function is applied to remove the grouping. Red arrows and labels indicate the flow of data and operations:

- verb**: Points to the `group_by` and `ungroup` functions.
- variable to group by**: Points to the `allstar_break` argument of the `group_by` function.
- verb**: Points to the `summarise` function.
- new variable**: Points to the `wins_perc` variable being created.
- function to summarize by**: Points to the `mean` function.
- remove grouping**: Points to the `ungroup` function.

ggplot2

```
allstar.plot = ggplot(data_figs_sum,  
                      aes(x = allstar_break,  
                           y = wins_perc))
```

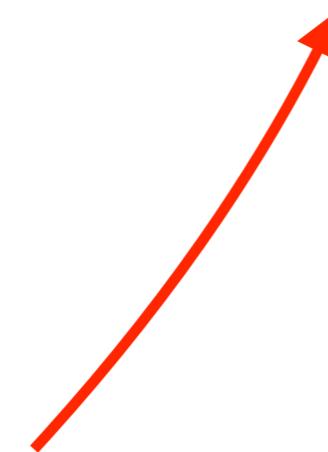
ggplot2

```
allstar.plot = ggplot(data_figs_sum,  
                      aes(x = allstar_break,  
                           y = wins_perc)) +
```

ggplot2

```
allstar.plot = ggplot(data_figs_sum,  
                      aes(x = allstar_break,  
                           y = wins_perc)) +  
  geom_bar()  
  
```

plot
type



ggplot2

```
allstar.plot = ggplot(data_figs_sum,  
                      aes(x = allstar_break,  
                           y = wins_perc)) +  
  geom_bar(stat = "identity")
```

plot type
method
of plotting
bars

ggplot2

```
allstar.plot = ggplot(data_figs_sum,  
                      aes(x = allstar_break,  
                           y = wins_perc)) +  
  geom_bar(stat = "identity")
```

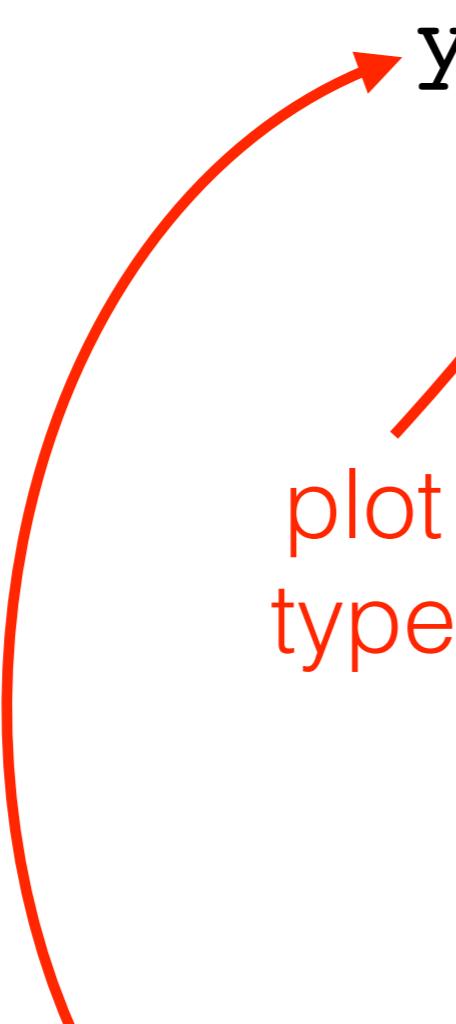
plot
type

method
of plotting
bars

use numbers
we computed

ggplot2

```
allstar.plot = ggplot(data_figs_sum,  
                      aes(x = allstar_break,  
                           y = wins_perc)) +  
  geom_bar(stat = "identity") +
```



plot type

method
of plotting
bars

use numbers
we computed

scale for the
y-axis

0, 100

0, 100

0, 100

0, 100