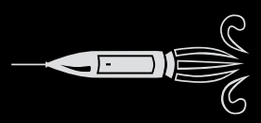
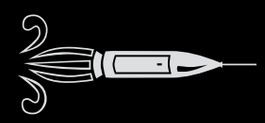
W14 — Introduction to Modern Statistics* with R

Dr. Shawn Cokus

October 22/23/24, 2024 — 1:30 to 4:30PM



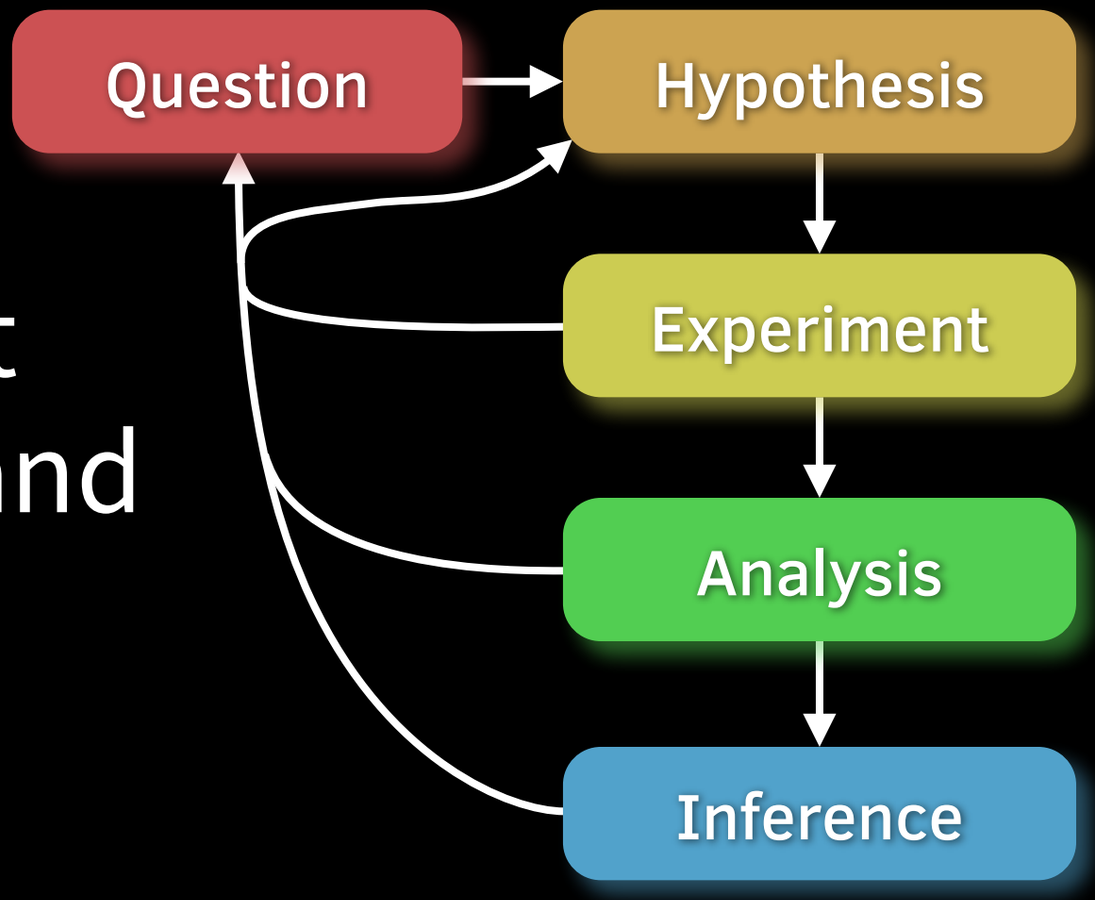
PART ONE



***QUIZ and HOMEWORK at end of Workshop is required for credit**, recommended for all attendees*

**and some "data science" maybe*

Scientific Research

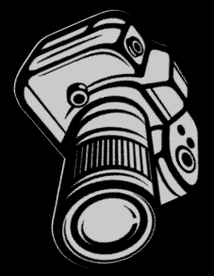


- **Statistics** is important part of the **Scientific Method** and dealing with experiments
  - ***Descriptive statistics:*** summarize and understand
  - ***Inferential statistics:*** draw conclusions
  - ***Exploratory data analysis:*** form hypotheses, select statistical tools/techniques, check assumptions

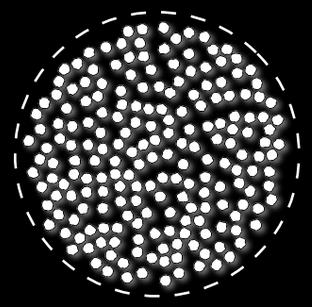
- *Subject too large to give comprehensive overview in the time we have!*

  WORKSHOP: review some basics, dig into R, and go through some hopefully useful ideas/examples

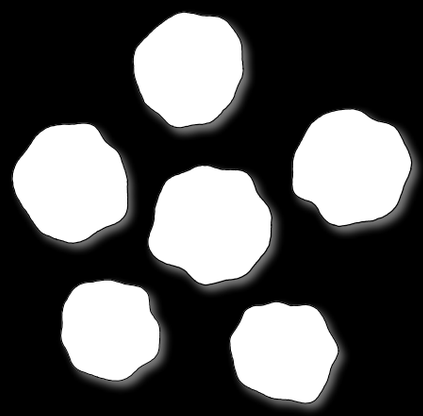
A Common Type of Experiment



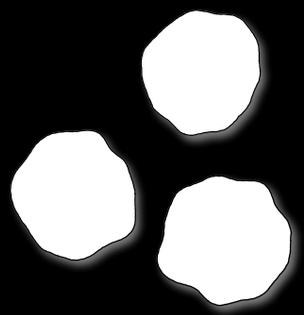
Essentially infinite POPULATION



RANDOM SAMPLE

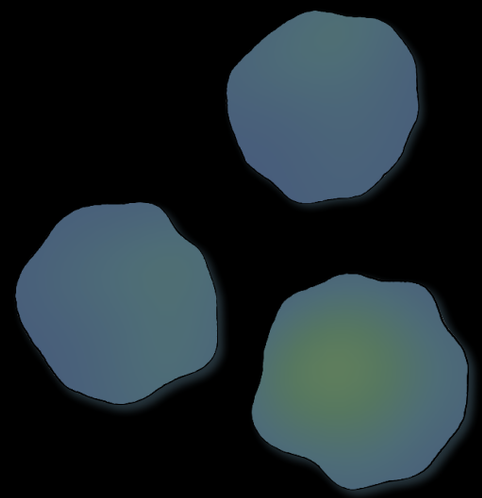


RANDOM ASSIGN

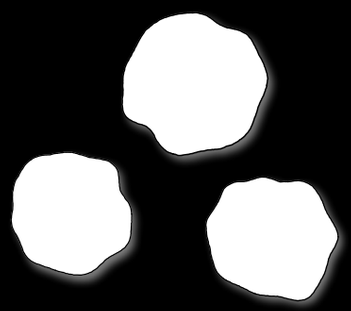


∅

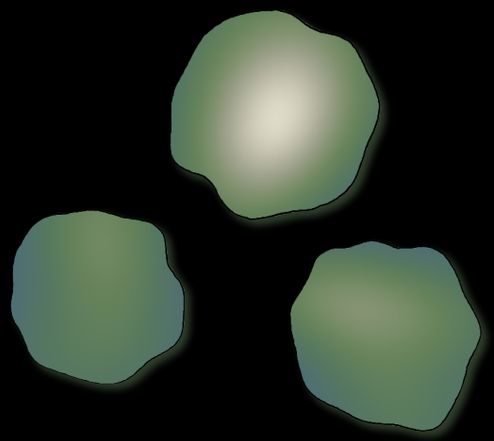
CONTROL



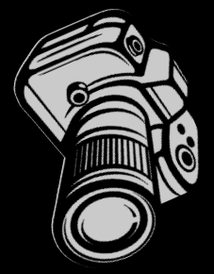
OBSERVED OUTCOMES

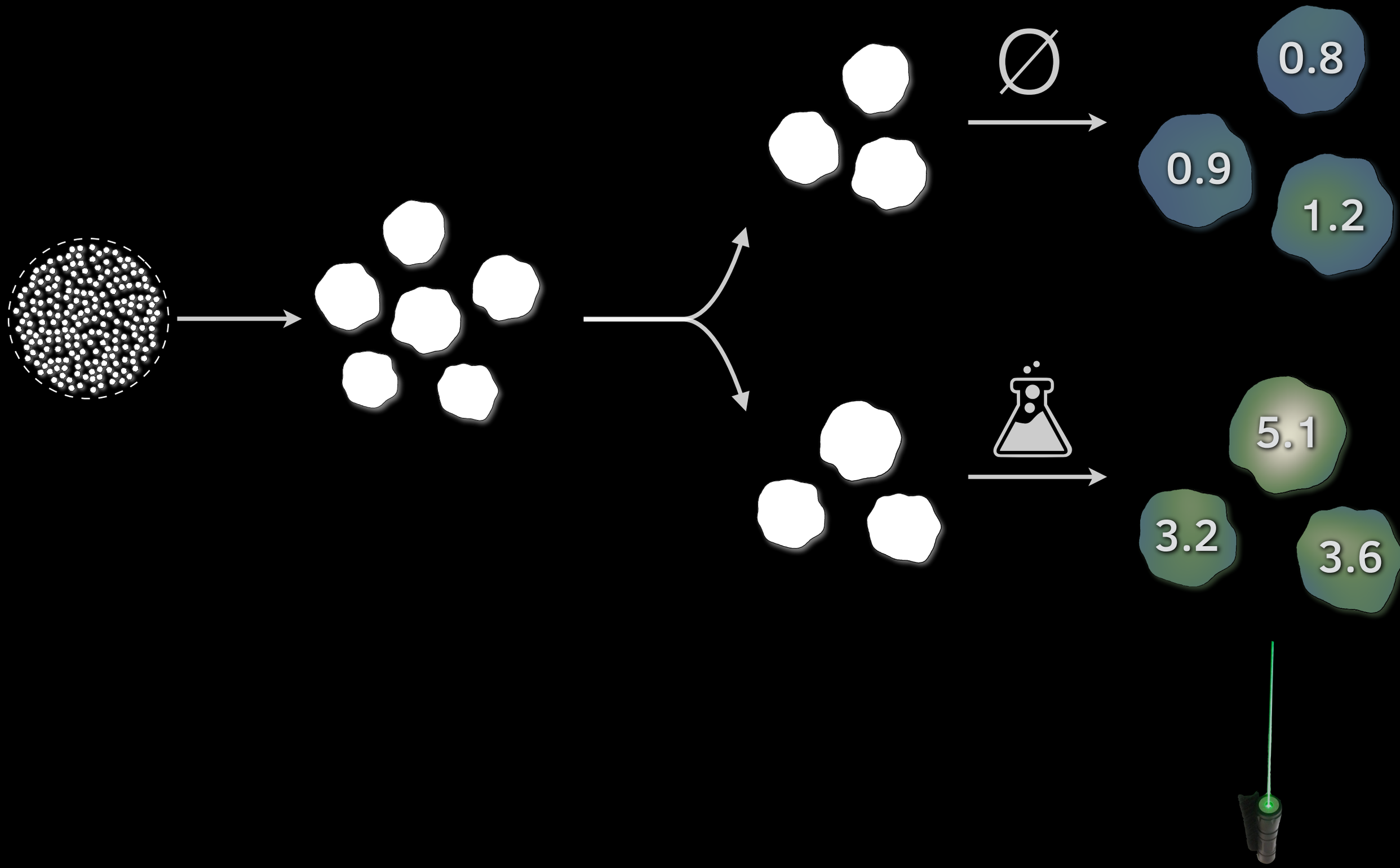


TREATED



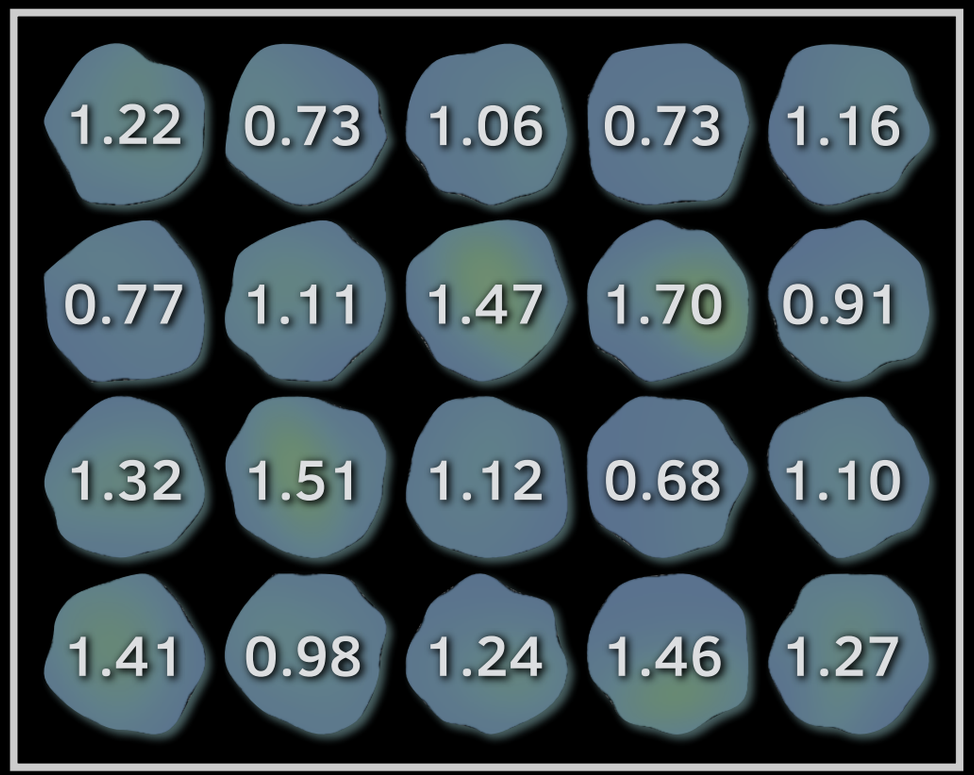
Quantitative Observations

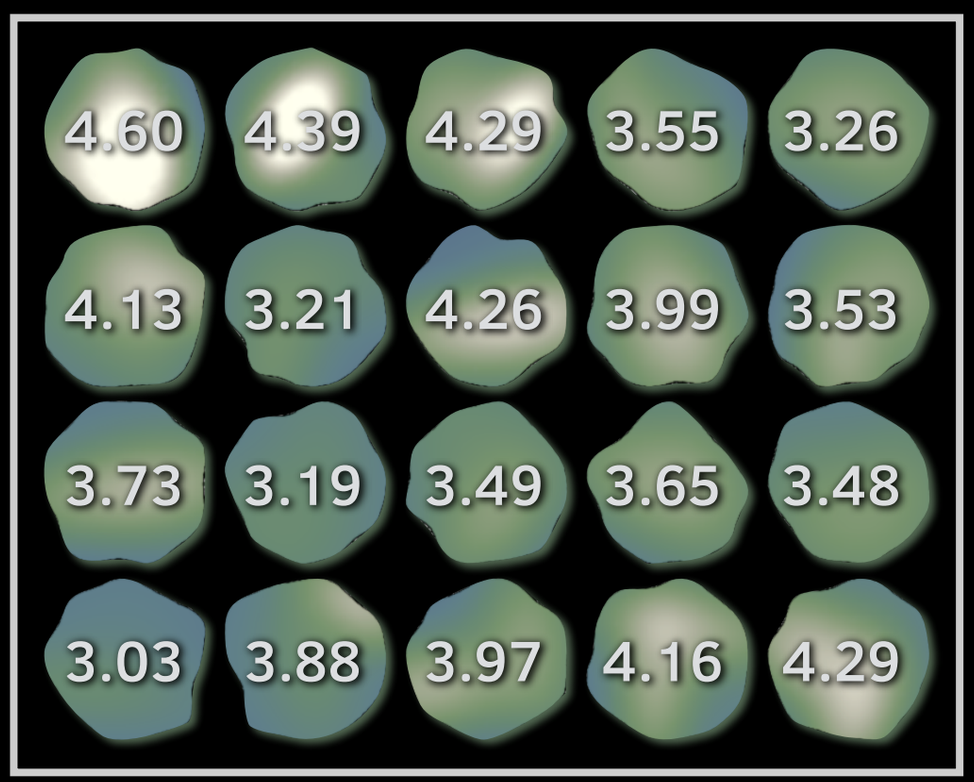




Increased Sample Sizes

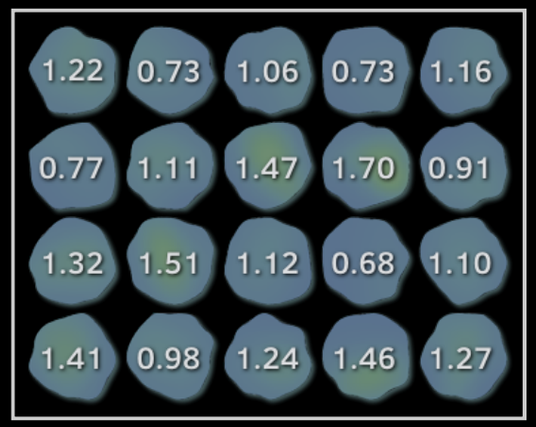
∅

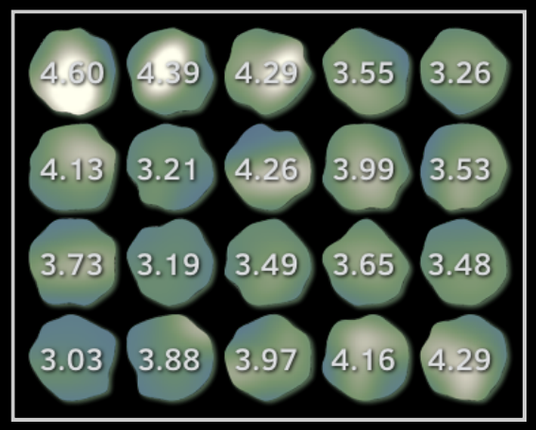




Summarizing...

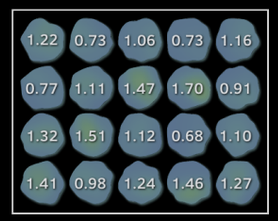
∅

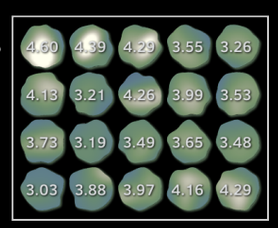




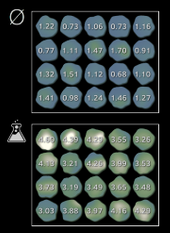
Summarizing...

∅

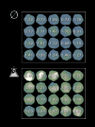




Summarizing…

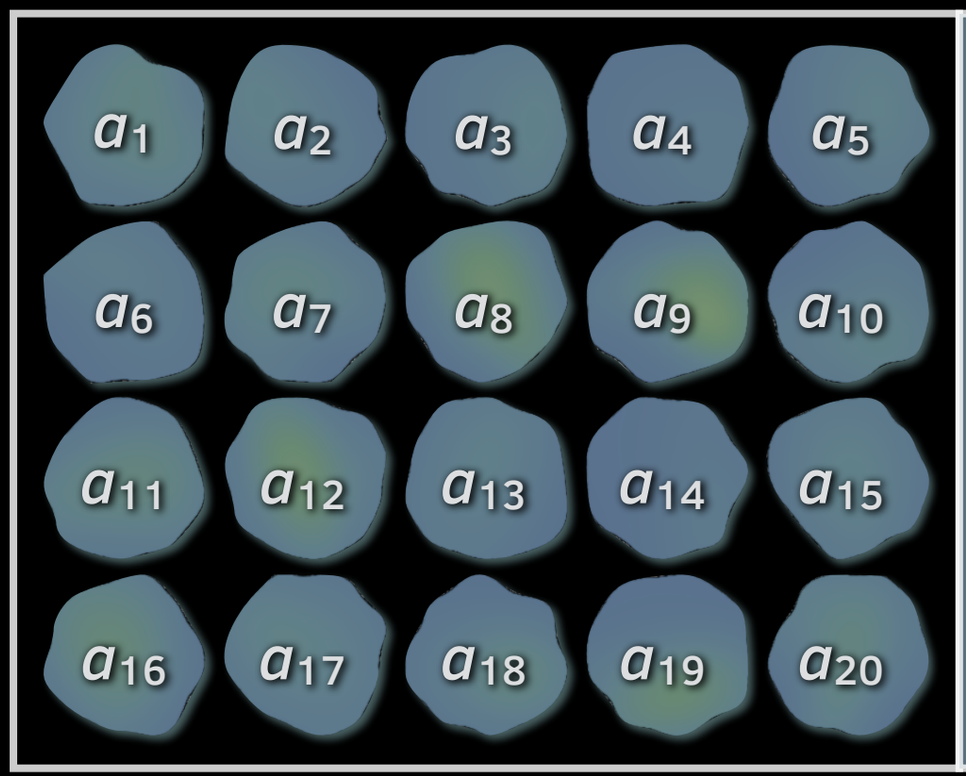


Summarizing...

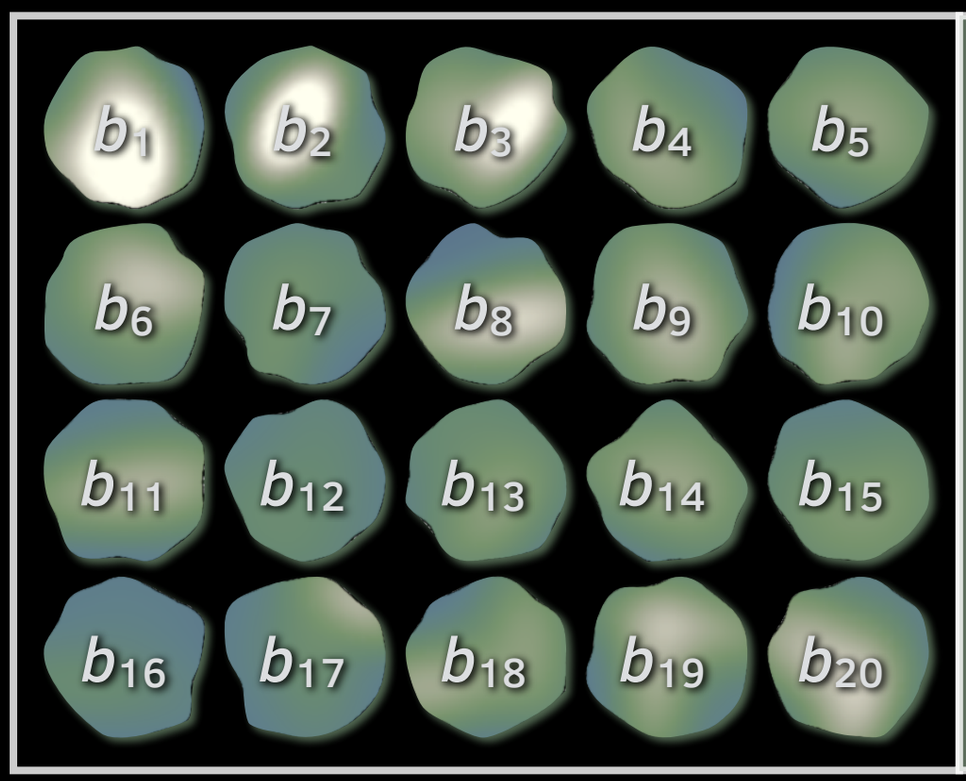


Summarizing...

Summarizing...



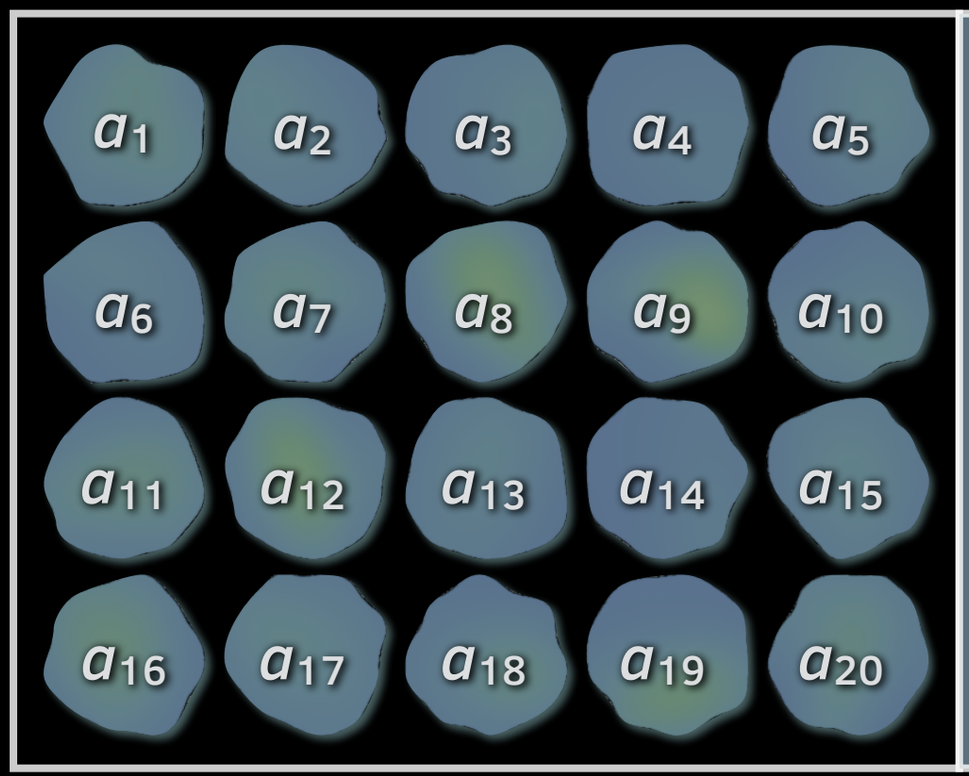
$$a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15} + a_{16} + a_{17} + a_{18} + a_{19} + a_{20}$$



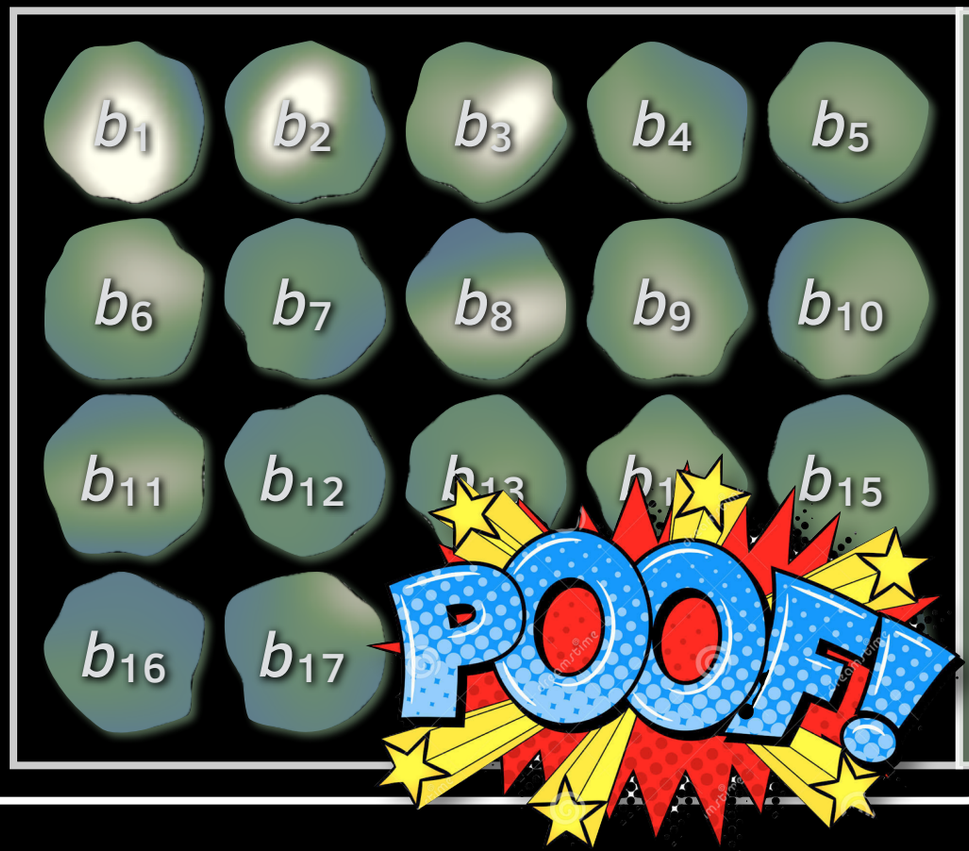
$$b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + b_{15} + b_{16} + b_{17} + b_{18} + b_{19} + b_{20}$$

… sometimes some replicates lost …

$\emptyset$



$$a_1 + a_2 + a_3 + a_4 + a_5$$
$$+ a_6 + a_7 + a_8 + a_9 + a_{10}$$
$$+ a_{11} + a_{12} + a_{13} + a_{14} + a_{15}$$
$$+ a_{16} + a_{17} + a_{18} + a_{19} + a_{20}$$

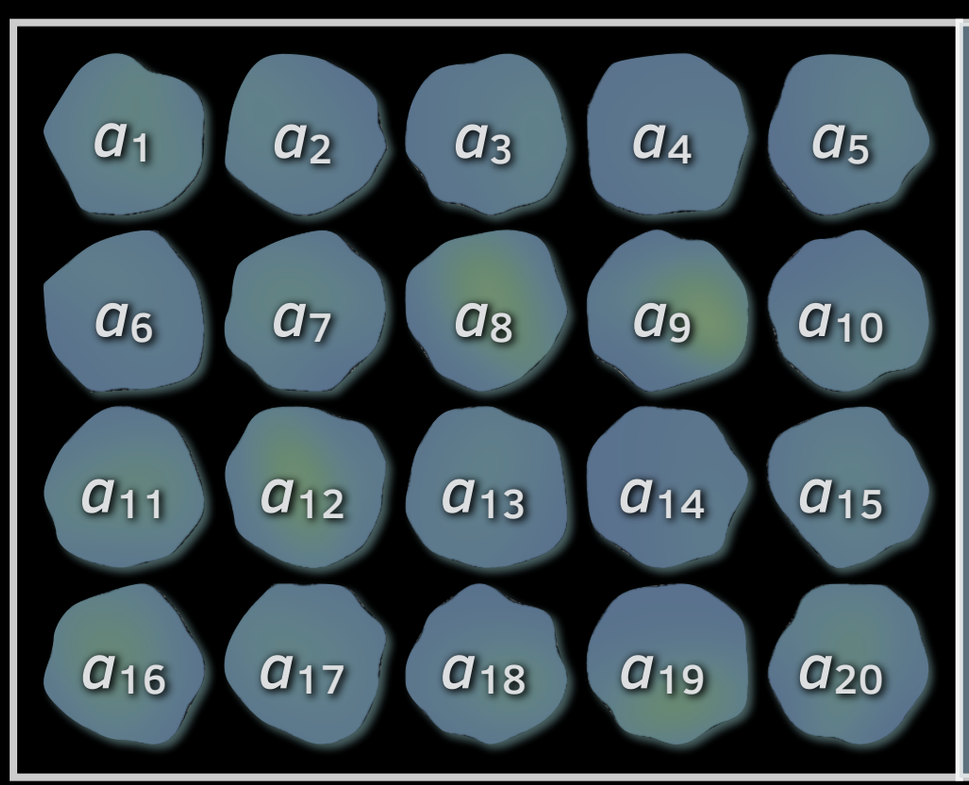


$$b_1 + b_2 + b_3 + b_4 + b_5$$
$$+ b_6 + b_7 + b_8 + b_9 + b_{10}$$
$$+ b_{11} + b_{12} + b_{13} + b_{14} + b_{15}$$
$$+ b_{16} + b_{17}$$

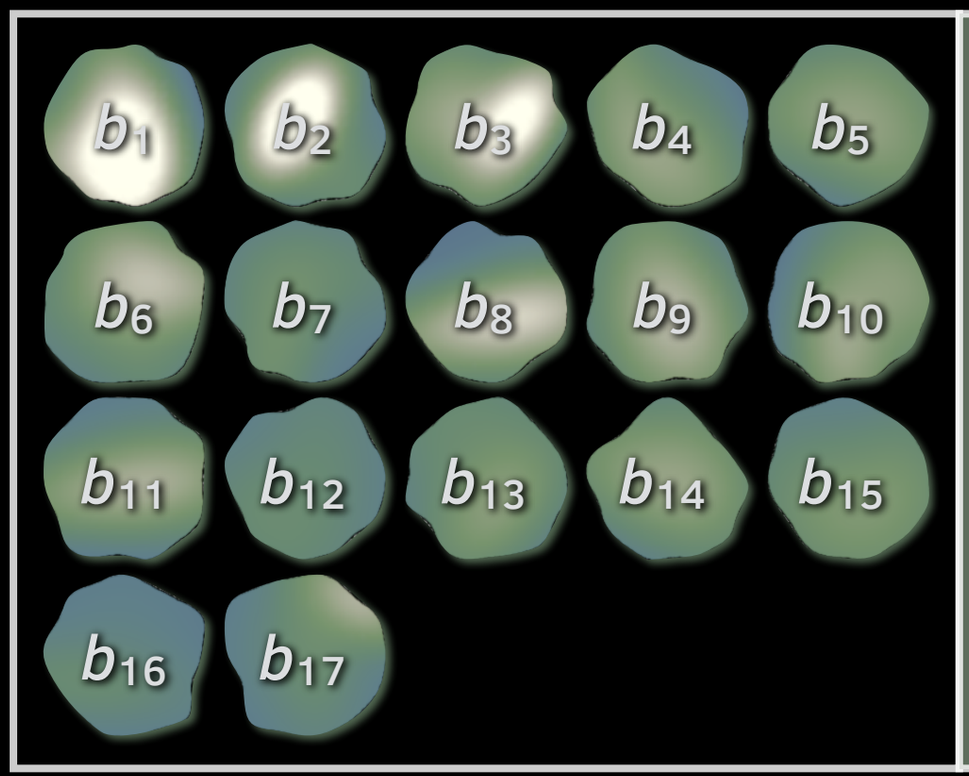
POOF!

… sometimes some replicates lost …

∅

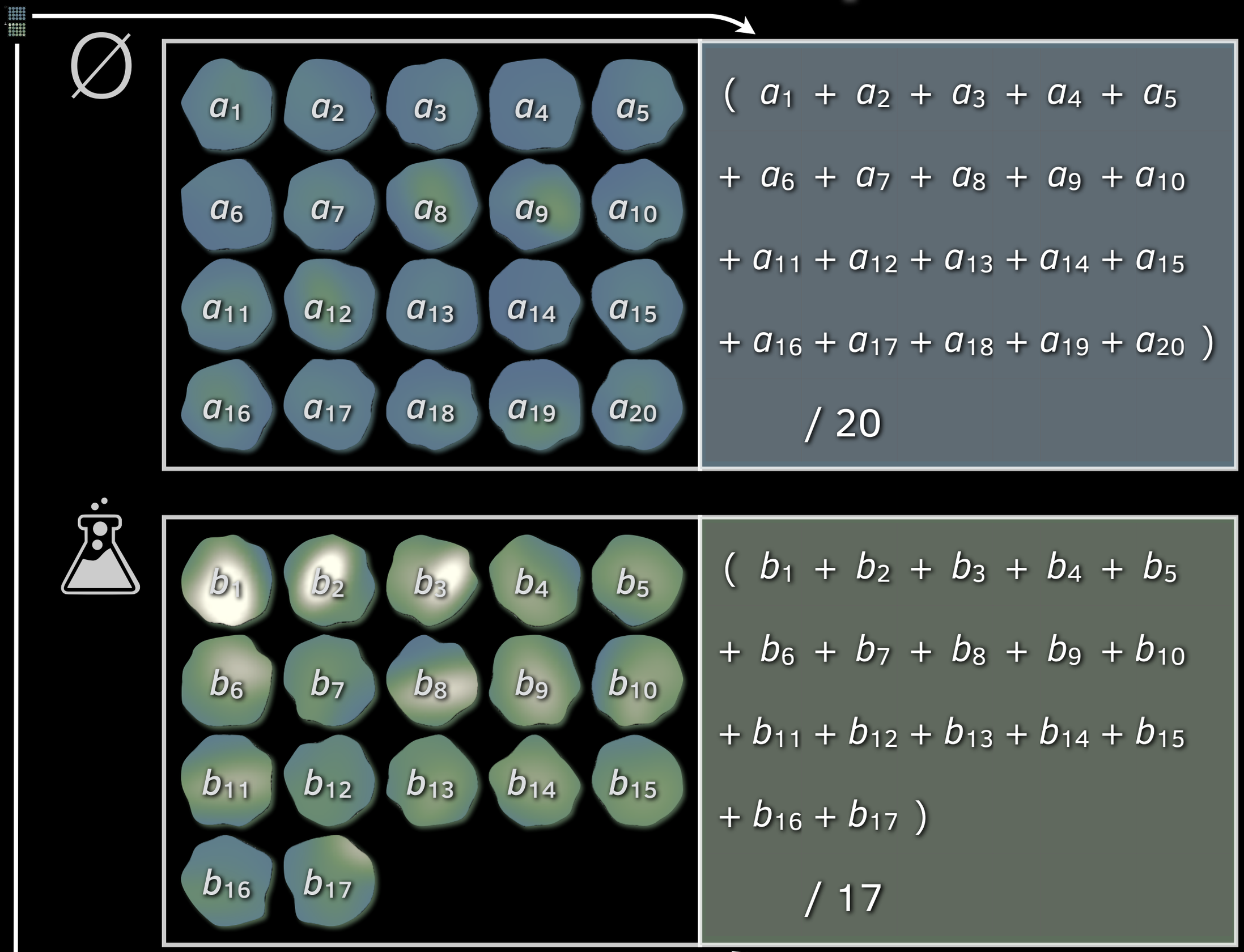


$$a_1 + a_2 + a_3 + a_4 + a_5$$
$$+ a_6 + a_7 + a_8 + a_9 + a_{10}$$
$$+ a_{11} + a_{12} + a_{13} + a_{14} + a_{15}$$
$$+ a_{16} + a_{17} + a_{18} + a_{19} + a_{20}$$



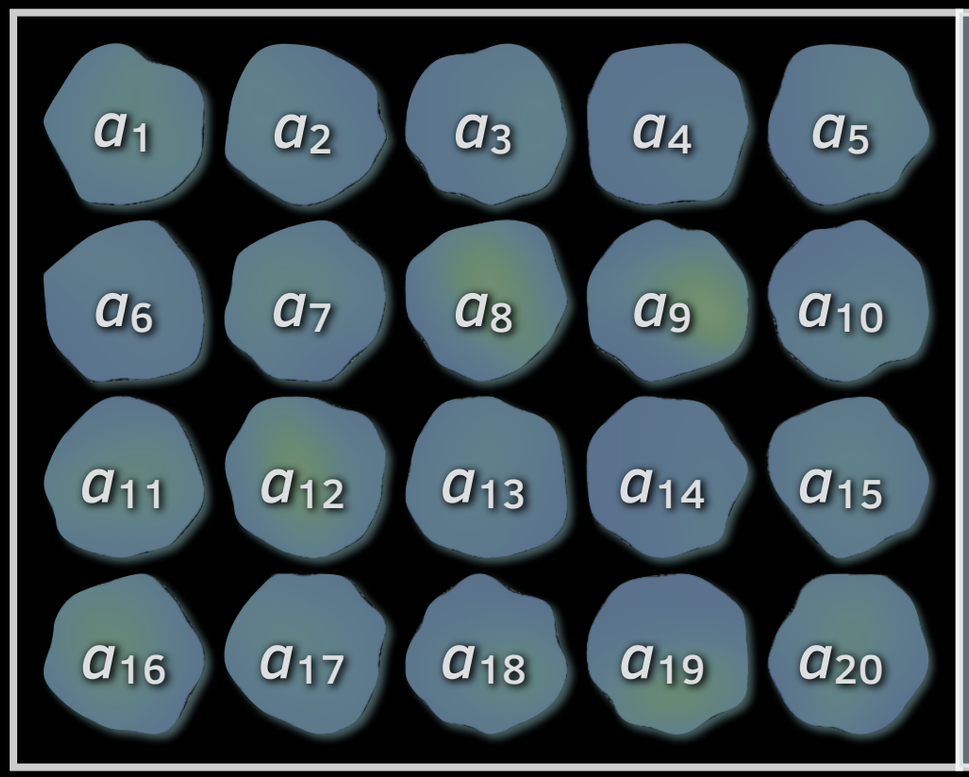
$$b_1 + b_2 + b_3 + b_4 + b_5$$
$$+ b_6 + b_7 + b_8 + b_9 + b_{10}$$
$$+ b_{11} + b_{12} + b_{13} + b_{14} + b_{15}$$
$$+ b_{16} + b_{17}$$

… divide so less size-dependent …



DESCRIPTIVE STAT: Arithmetic MEAN

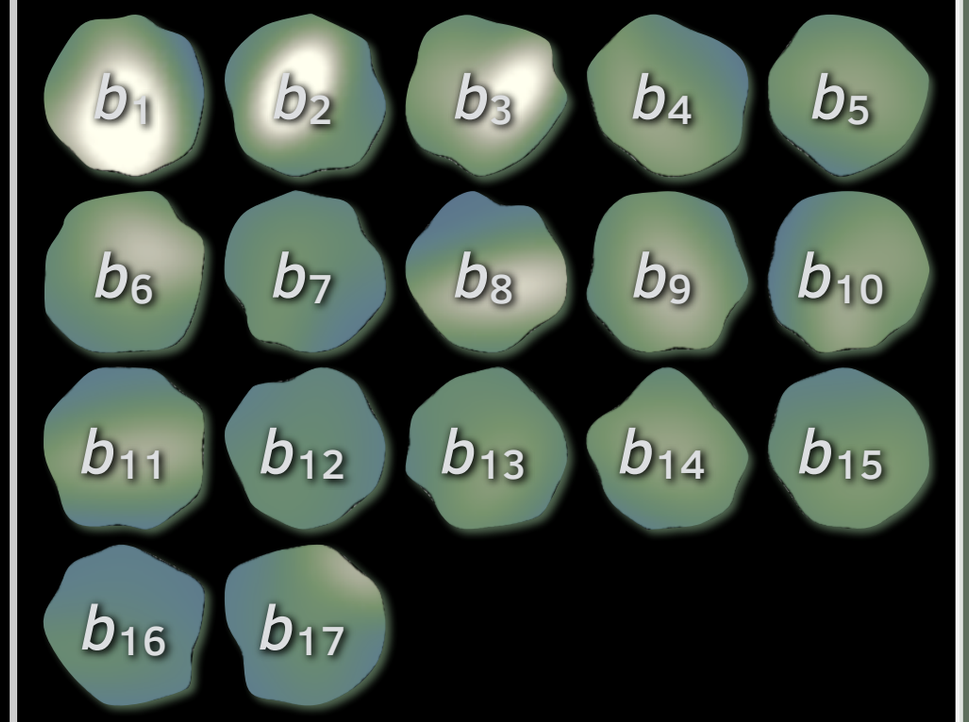
∅



$$(a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15} + a_{16} + a_{17} + a_{18} + a_{19} + a_{20})$$

$$/ 20: \quad \text{“}\overline{a_{1..20}}\text{”} \approx \mathbf{1.15}$$

SAMPLE MEAN

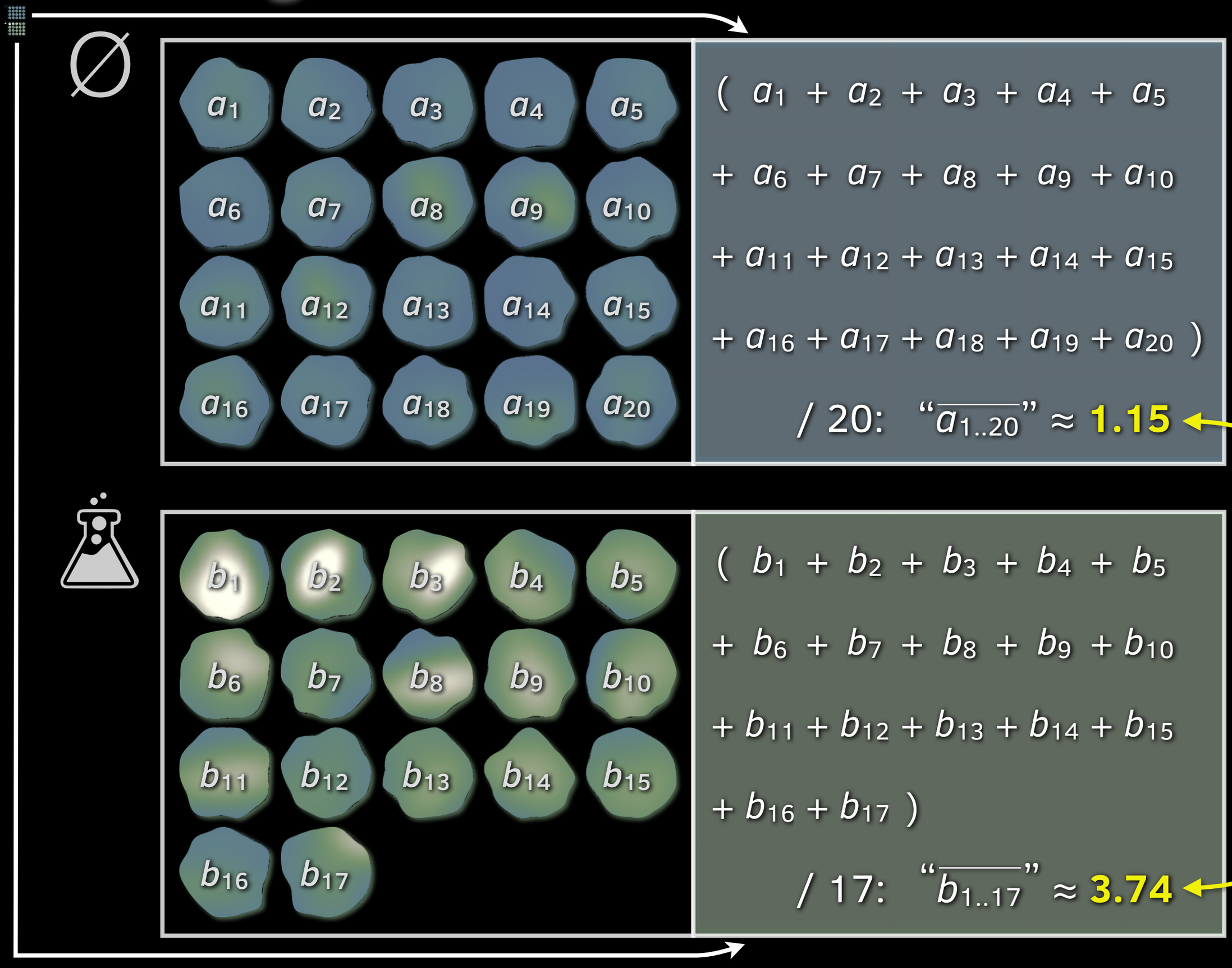


$$(b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + b_{15} + b_{16} + b_{17})$$

$$/ 17: \quad \text{“}\overline{b_{1..17}}\text{”} \approx \mathbf{3.74}$$

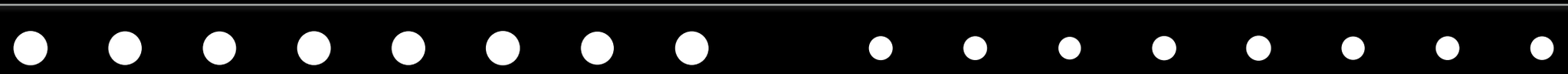
SAMPLE MEAN

Working Toward Inference…



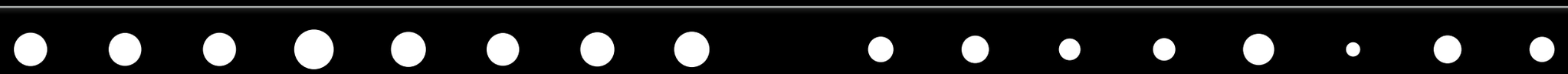
A Game: Guess the Truth — Average area of population LEFT SIDE came from is **< > =** that of population RIGHT SIDE came from?

Game 1: 

Game 2: 

Game 3: 

Game 4: 

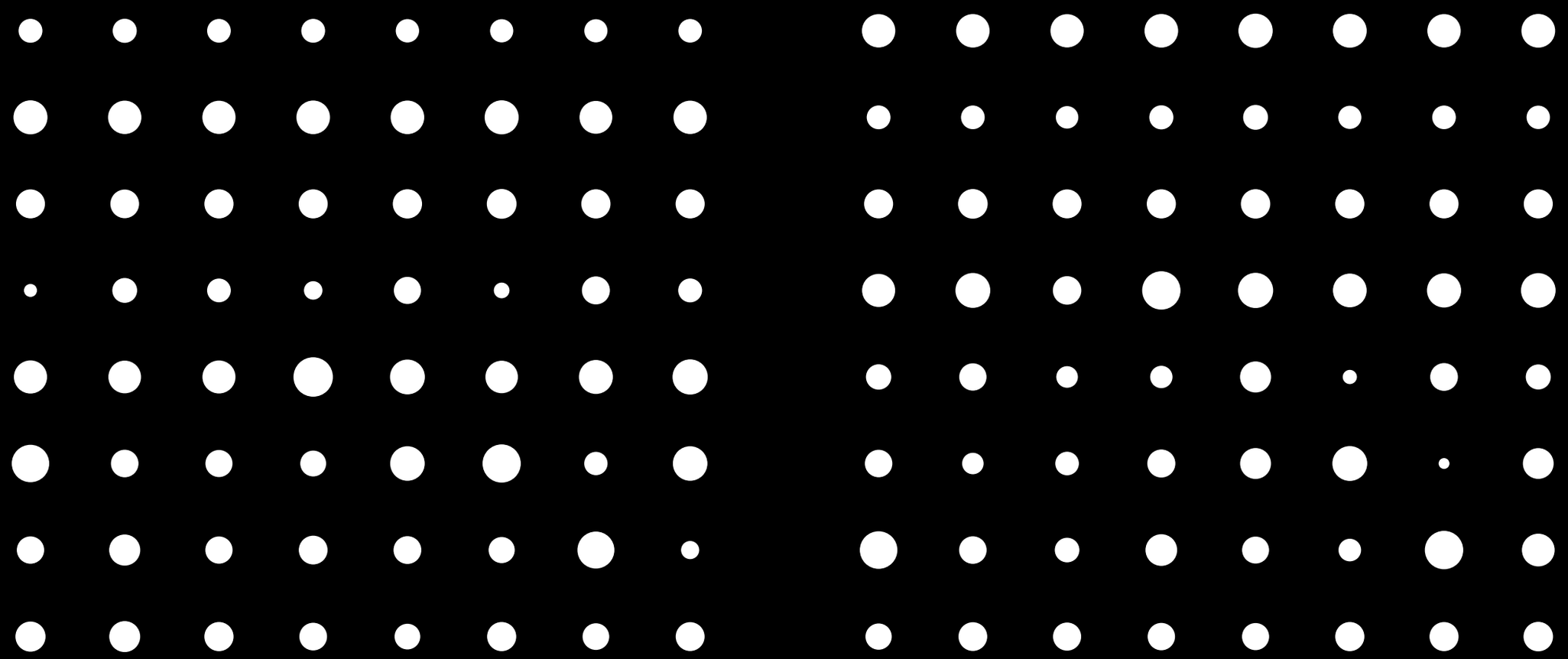
Game 5: 

Game 6: 

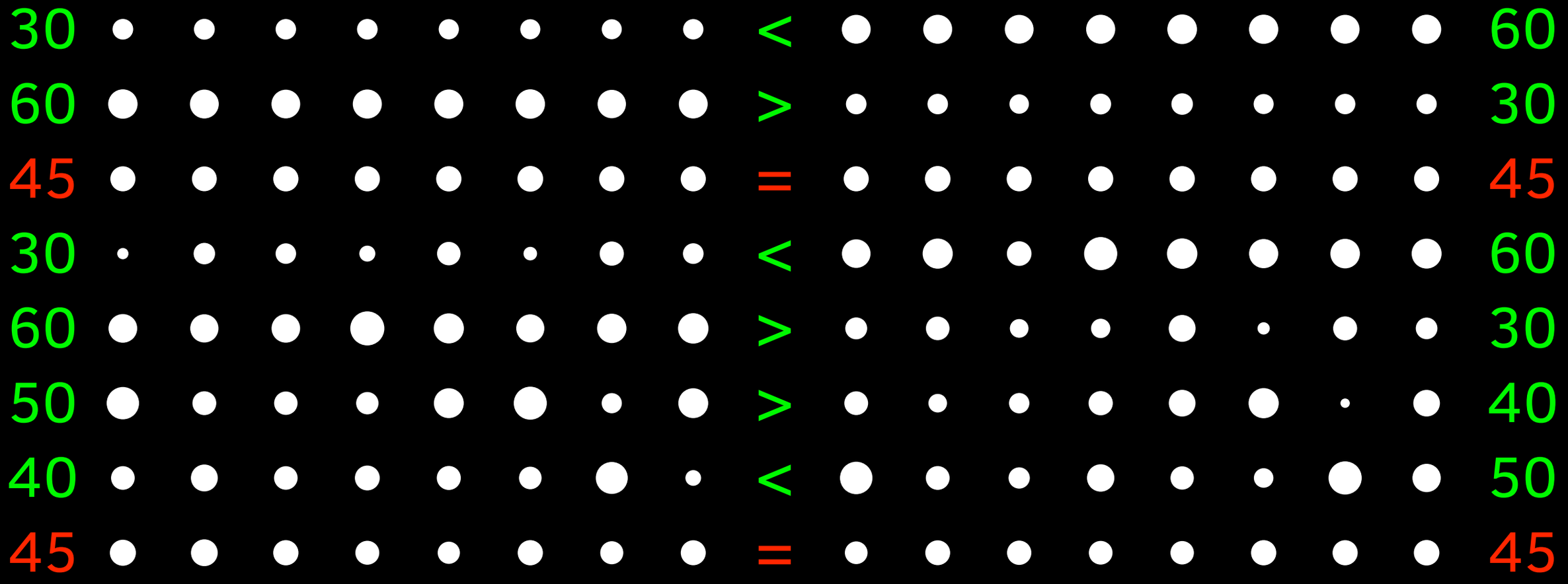
Game 7: 

Game 8: 

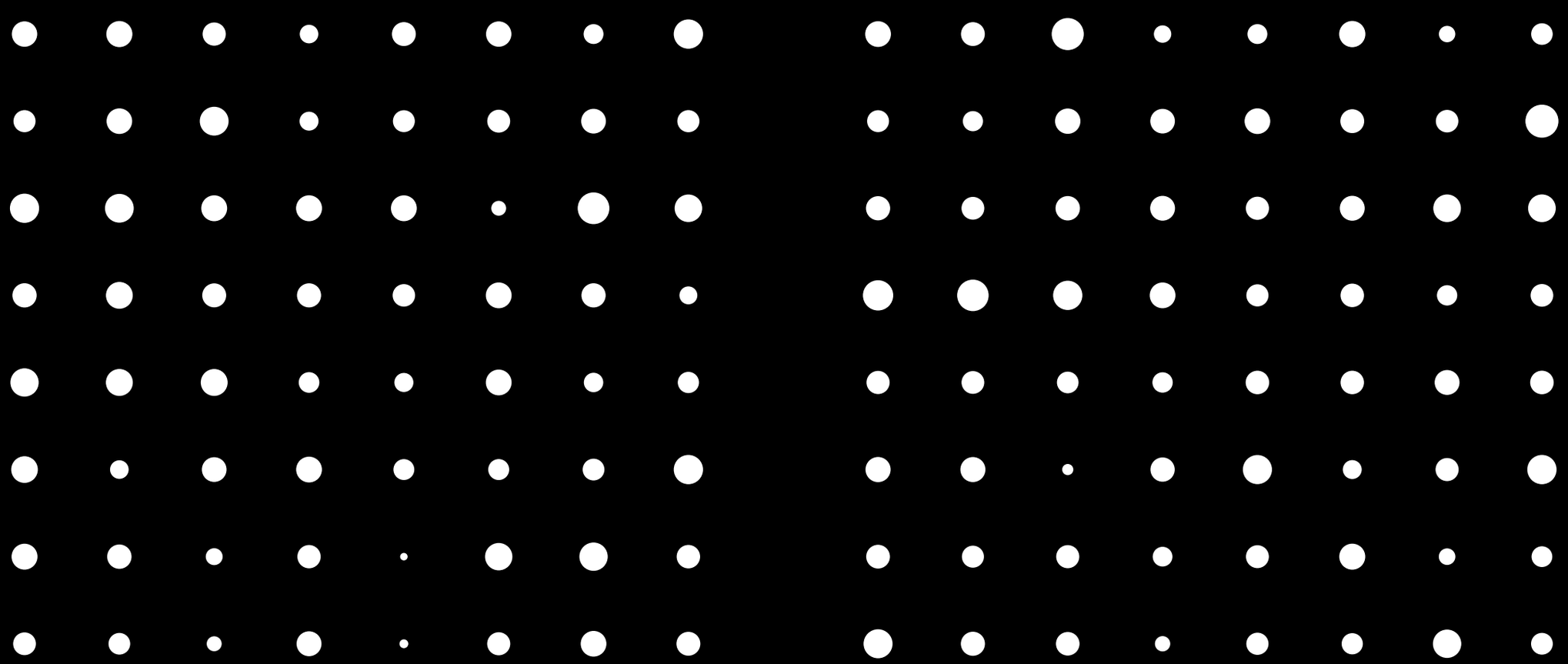
A Game: Guess the Truth — Average area of population LEFT SIDE came from is **< > =** that of population RIGHT SIDE came from?



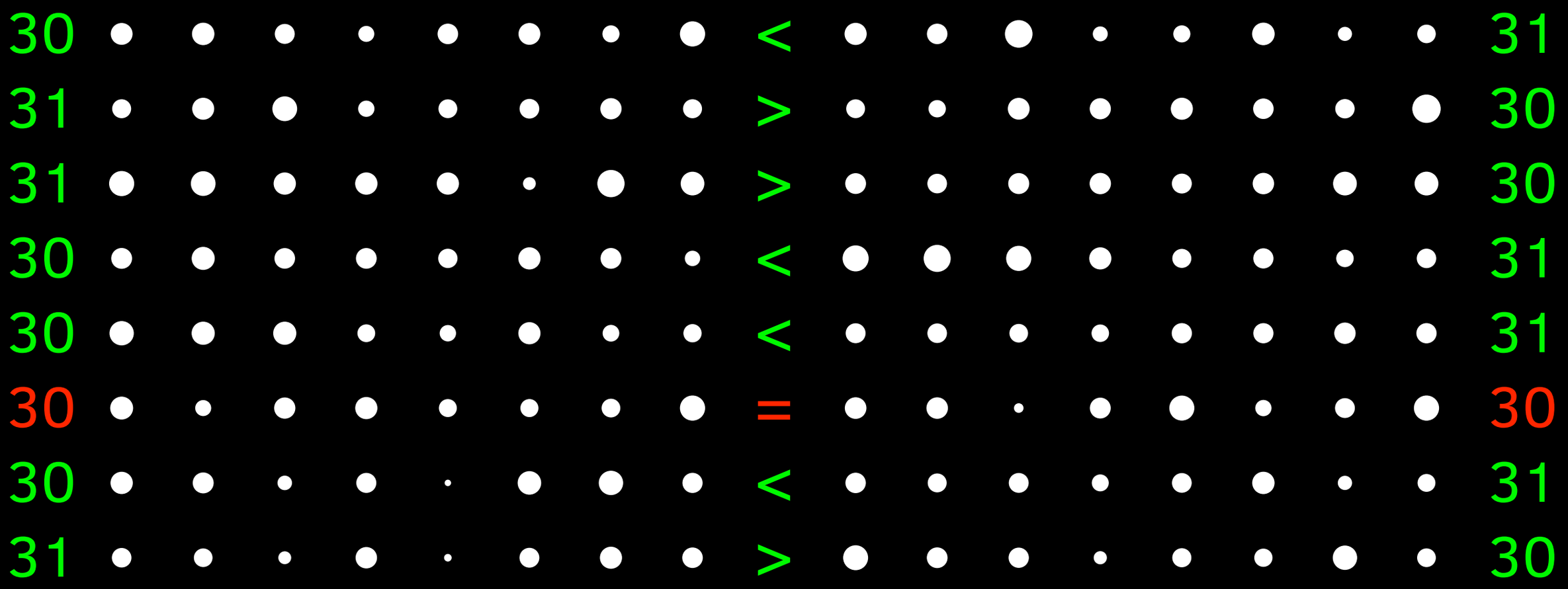
A Game: Guess the Truth — Average area of population LEFT SIDE came from is **< > =** that of population RIGHT SIDE came from?



Harder: Guess the Truth — Average area of population LEFT SIDE came from is **< > =** that of population RIGHT SIDE came from?



Harder: Guess the Truth — Average area of population LEFT SIDE came from is **< > =** that of population RIGHT SIDE came from?



Quantifying Variation

∅

individual observations

1.22	0.73	1.06	0.73	1.16
0.77	1.11	1.47	1.70	0.91
1.32	1.51	1.12	0.68	1.10
1.41	0.98	1.24	1.46	1.27

average

$$(a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15} + a_{16} + a_{17} + a_{18} + a_{19} + a_{20})$$

$$/ 20: \quad \overline{a_{1..20}} \approx \mathbf{1.15}$$

individual observations

4.60	4.39	4.29	3.55	3.26
4.13	3.21	4.26	3.99	3.53
3.73	3.19	3.49	3.65	3.48
3.03	3.88			

average

$$(b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + b_{15} + b_{16} + b_{17})$$

$$/ 17: \quad \overline{b_{1..17}} \approx \mathbf{3.74}$$

Quantifying Variation

∅

differences from average

+.07	−.42	−.09	−.42	+.01
−.38	−.04	+.32	+.55	−.24
+.17	+.36	−.03	−.47	−.05
+.26	−.17	+.09	+.31	+.12

average

$$(a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15} + a_{16} + a_{17} + a_{18} + a_{19} + a_{20})$$

$$/ 20: \quad \overline{a_{1..20}} \approx \mathbf{1.15}$$

differences from average

+.86	+.65	+.55	−.19	−.48
+.39	−.53	+.52	+.25	−.21
−.01	−.55	−.25	−.09	−.26
−.71	+.14			

average

$$(b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + b_{15} + b_{16} + b_{17})$$

$$/ 17: \quad \overline{b_{1..17}} \approx \mathbf{3.74}$$

Quantifying Variation

∅

squared differences

$.07^2$	$.42^2$	$.09^2$	$.42^2$	$.01^2$
$.38^2$	$.04^2$	$.32^2$	$.55^2$	$.24^2$
$.17^2$	$.36^2$	$.03^2$	$.47^2$	$.05^2$
$.26^2$	$.17^2$	$.09^2$	$.31^2$	$.12^2$

average

$$(a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15} + a_{16} + a_{17} + a_{18} + a_{19} + a_{20})$$

$$/ 20: \quad \overline{a_{1..20}} \approx \mathbf{1.15}$$

squared differences

$.86^2$	$.65^2$	$.55^2$	$.19^2$	$.48^2$
$.39^2$	$.53^2$	$.52^2$	$.25^2$	$.21^2$
$.01^2$	$.55^2$	$.25^2$	$.09^2$	$.26^2$
$.71^2$	$.14^2$			

average

$$(b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + b_{15} + b_{16} + b_{17})$$

$$/ 17: \quad \overline{b_{1..17}} \approx \mathbf{3.74}$$

Summarize Differences²? Try mean…

∅

average squared differences

$.07^2$ $.42^2$ $.09^2$ $.42^2$ $.01^2$

$.38^2$ $.\_^2$ $.32^2$ $.55^2$ $.24^2$

$\sum$ / 20

$.17^2$ $.\_^2$ $.\_^2$ $.\_^2$ $.05^2$

$.26^2$ $.17^2$ $.09^2$ $.31^2$ $.12^2$

average

$$(a_1 + a_2 + a_3 + a_4 + a_5$$

$$+ a_6 + a_7 + a_8 + a_9 + a_{10}$$

$$+ a_{11} + a_{12} + a_{13} + a_{14} + a_{15}$$

$$+ a_{16} + a_{17} + a_{18} + a_{19} + a_{20})$$

$$/ 20: \quad \overline{a_{1..20}} \approx \mathbf{1.15}$$

average squared differences

$.86^2$ $.65^2$ $.55^2$ $.19^2$ $.48^2$

$.39^2$ $.\_^2$ $.52^2$ $.25^2$ $.21^2$

$\sum$ / 17

$.01^2$ $.\_^2$ $.\_^2$ $.09^2$ $.26^2$

$.71^2$ $.14^2$

average

$$(b_1 + b_2 + b_3 + b_4 + b_5$$

$$+ b_6 + b_7 + b_8 + b_9 + b_{10}$$

$$+ b_{11} + b_{12} + b_{13} + b_{14} + b_{15}$$

$$+ b_{16} + b_{17})$$

$$/ 17: \quad \overline{b_{1..17}} \approx \mathbf{3.74}$$

DESCRIPTIVE STATISTIC: VARIANCE

∅

average squared differences

$$\frac{\sum_{i=1}^{20} (a_i - \bar{\mu}_a)^2}{20}$$

$\approx$ **0.08** $=: \bar{\sigma}^2_a$

SAMPLE* VARIANCE

average

$$(a_1 + a_2 + a_3 + a_4 + a_5$$

$$+ a_6 + a_7 + a_8 + a_9 + a_{10}$$

$$+ a_{11} + a_{12} + a_{13} + a_{14} + a_{15}$$

$$+ a_{16} + a_{17} + a_{18} + a_{19} + a_{20})$$

/ 20: $\overline{a_{1..20}} \approx$ **1.15** $=: \bar{\mu}_a$

average squared differences

$$\frac{\sum_{i=1}^{17} (b_i - \bar{\mu}_b)^2}{17}$$

$\approx$ **0.21** $=: \bar{\sigma}^2_b$

SAMPLE* VARIANCE

average

$$(b_1 + b_2 + b_3 + b_4 + b_5$$

$$+ b_6 + b_7 + b_8 + b_9 + b_{10}$$

$$+ b_{11} + b_{12} + b_{13} + b_{14} + b_{15}$$

$$+ b_{16} + b_{17})$$

/ 17: $\overline{b_{1..17}} \approx$ **3.74** $=: \bar{\mu}_b$

DESCRIPTIVE STATISTIC: VARIANCE

∅

average squared differences

$$\frac{\sum_{i=1}^{20}(a_i - \bar{\mu}_a)^2}{20}$$

$$\approx 0.08 =: \bar{\sigma}^2{}_a$$

SAMPLE* VARIANCE

(∑ (int.–int.)²) / 20:

INTENSITY²

average

$$(\ a_1 + a_2 + a_3 + a_4 + a_5$$

$$+ a_6 + a_7 + a_8 + a_9 + a_{10}$$

$$+ a_{11} + a_{12} + a_{13} + a_{14} + a_{15}$$

$$+ a_{16} + a_{17} + a_{18} + a_{19} + a_{20}\ )$$

$$/\ 20: \quad 4.15 =: \bar{\mu}_a$$

(∑ intensity) / 20:

INTENSITY

average squared differences

$$\frac{\sum_{i=1}^{17}(b_i - \bar{\mu}_b)^2}{17}$$

$$\approx 0.21 =: \bar{\sigma}^2{}_b$$

SAMPLE* VARIANCE

(∑ (int.–int.)²) / 17:

INTENSITY²

average

$$(\ b_1 + b_2 + b_3 + b_4 + b_5$$

$$+ b_6 + b_7 + b_8 + b_9 + b_{10}$$

$$+ b_{11} + b_{12} + b_{13} + b_{14} + b_{15}$$

$$+ b_{16} + b_{17}\ )$$

$$/\ 17: \quad 3.74 =: \bar{\mu}_b$$

(∑ intensity) / 17:

INTENSITY

STATISTIC: STANDARD DEVIATION

∅

average squared differences

$$\frac{\sum_{i=1}^{20}(a_i - \bar{\mu}_a)^2}{20}$$

$$\approx 0.08 =: \bar{\sigma}^2{}_a$$

$$\bar{\sigma}_a := \sqrt{\bar{\sigma}^2{}_a} \approx 0.28$$

SAMPLE* STANDARD DEVIATION

average

$$(a_1 + a_2 + a_3 + a_4 + a_5$$

$$+ a_6 + a_7 + a_8 + a_9 + a_{10}$$

$$+ a_{11} + a_{12} + a_{13} + a_{14} + a_{15}$$

$$+ a_{16} + a_{17} + a_{18} + a_{19} + a_{20})$$

$$/ 20: \overline{a_{1..20}} \approx 1.15 =: \bar{\mu}_a$$

average squared differences

$$\frac{\sum_{i=1}^{17}(b_i - \bar{\mu}_b)^2}{17}$$

$$\approx 0.21 =: \bar{\sigma}^2{}_b$$

$$\bar{\sigma}_b := \sqrt{\bar{\sigma}^2{}_b} \approx 0.45$$

SAMPLE* STANDARD DEVIATION

average

$$(b_1 + b_2 + b_3 + b_4 + b_5$$

$$+ b_6 + b_7 + b_8 + b_9 + b_{10}$$

$$+ b_{11} + b_{12} + b_{13} + b_{14} + b_{15}$$

$$+ b_{16} + b_{17})$$

$$/ 17: \overline{b_{1..17}} \approx 3.74 =: \bar{\mu}_b$$

BASIC DESCRIPTIVE STATISTICS

Group	Sample size n	Sample mean $\bar{\mu}$	Sample* stdDev. $\bar{\sigma}$
∅ control	20	1.15	0.28
treated	17	3.74	0.45

```
R> ctrl <- c(1.22, 0.73, 1.06, 0.73, 1.16,
             0.77, 1.11, 1.47, 1.70, 0.91,
             1.32, 1.51, 1.12, 0.68, 1.10,
             1.41, 0.98, 1.24, 1.46, 1.27  );   # length(ctrl): 20
trtd <- c(4.60, 4.39, 4.29, 3.55, 3.26,
          4.13, 3.21, 4.26, 3.99, 3.53,
          3.73, 3.19, 3.49, 3.65, 3.48,
          3.03, 3.88                    );   # length(trtd): 17
sum(ctrl) / length(ctrl)                     # 1.1475
sum(trtd) / length(trtd)                     # 3.744706
(meanC <- mean(ctrl))                        # 1.1475
(meanT <- mean(trtd))                        # 3.744706
(varC  <- mean( (ctrl-meanC)^2 ))            # 0.07860875
(varT  <- mean( (trtd-meanT)^2 ))            # 0.2061896
(sdC   <- sqrt(varC))                        # 0.2803724
(sdT   <- sqrt(varT))                        # 0.4540811
```

BASIC DESCRIPTIVE STATISTICS

Group	Sample size n	Sample mean $\bar{\mu}$	Sample* stdDev. $\bar{\sigma}$
∅ control	20	1.15	0.28
treated	17	3.74	0.45

```
R> ctrl <- c(1.22, 0.73, 1.06, 0.73, 1.16,
             0.77, 1.11, 1.47, 1.70, 0.91,
             1.32, 1.51, 1.12, 0.68, 1.10,
             1.41, 0.98, 1.24, 1.46, 1.27  );   # length(ctrl): 20
trtd <- c(4.60, 4.39, 4.29, 3.55,
          4.13, 3.21, 4.26, 3.99,
          3.73, 3.19, 3.49, 3.65,
          3.03, 3.88
sum(ctrl) / length(ctrl)
sum(trtd) / length(trtd)
(meanC <- mean(ctrl))
(meanT <- mean(trtd))
(varC  <- mean( (ctrl-meanC)^2 ))
(varT  <- mean( (trtd-meanT)^2 ))
(sdC   <- sqrt(varC))
(sdT   <- sqrt(varT))
```

R has complex default rules for "invisibility" (not all evaluation results that make it to, e.g., the top level of the console get `print`'ed).

`invisible(`*expr*`)` sets the flag, `(`*expr*`)` with "extra" parens. clears

`(meanT <- mean(trtd))`
just saves us an extra `meanT` or `.Last.value` line to see result

Many Ways to Store/Organize Data

Group	Sample size n	Sample mean $\bar{\mu}$	Sample* stdDev. $\bar{\sigma}$
∅ control	20	1.15	0.28
 treated	17	3.74	0.45

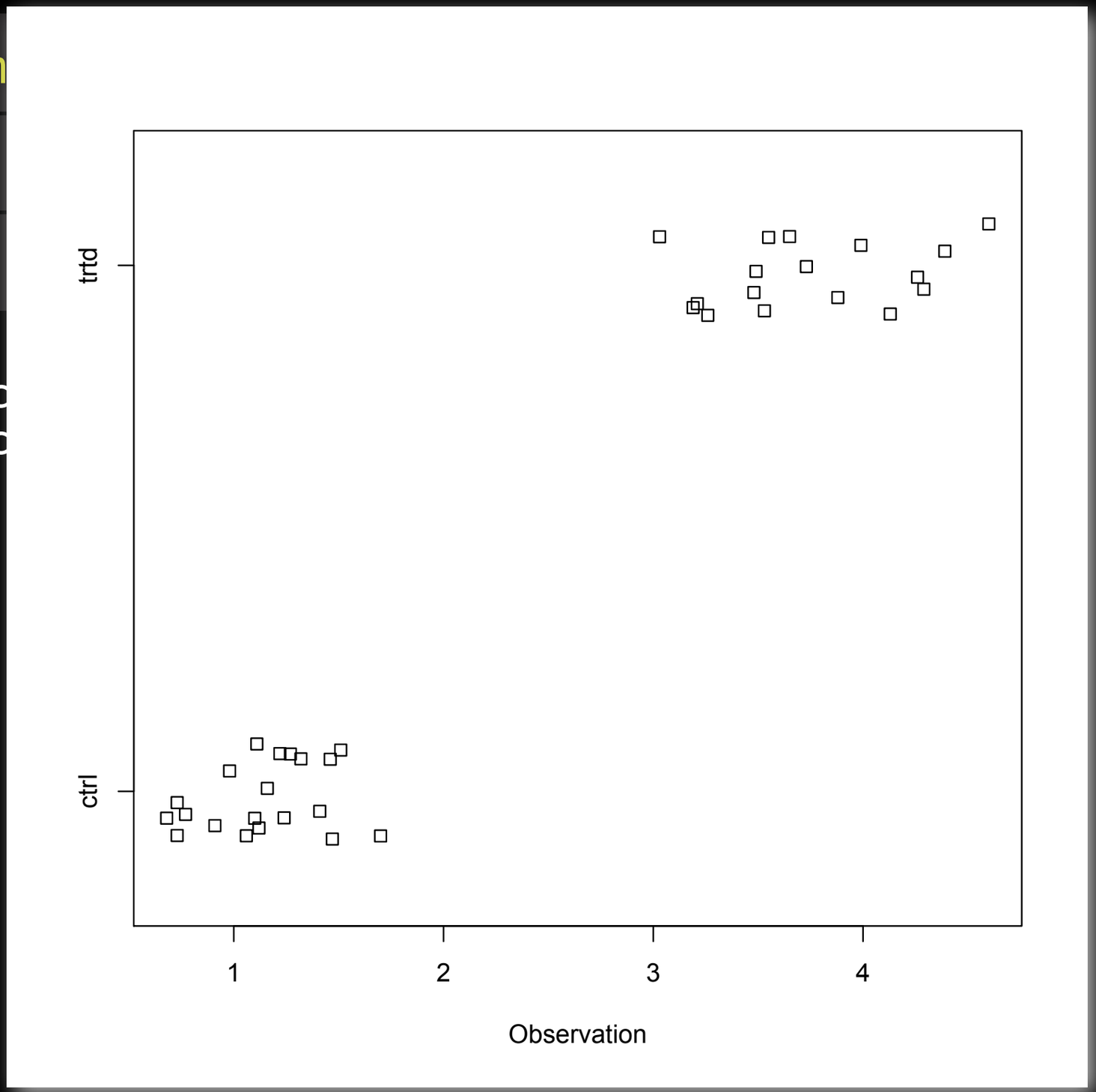
```
R> (df <- data.frame(obs=c(ctrl, trtd),
                     grp=c(rep("ctrl", length(ctrl)),
                           rep("trtd", length(trtd)) )))
   obs  grp
1  1.22 ctrl
2  0.73 ctrl
…… …… ……
19 1.46 ctrl
20 1.27 ctrl
21 4.60 trtd
22 4.39 trtd
…… …… ……
36 3.03 trtd
37 3.88 trtd
```

Always Explore/Visualize Your Data

Group	Sam
∅ control	
treated	

```
R> (df <- data.frame(obs=c
                     grp=c
```

```
    obs  grp
1  1.22 ctrl
2  0.73 ctrl
…… …… ……
19 1.46 ctrl
20 1.27 ctrl
21 4.60 trtd
22 4.39 trtd
…… …… ……
36 3.03 trtd
37 3.88 trtd
```



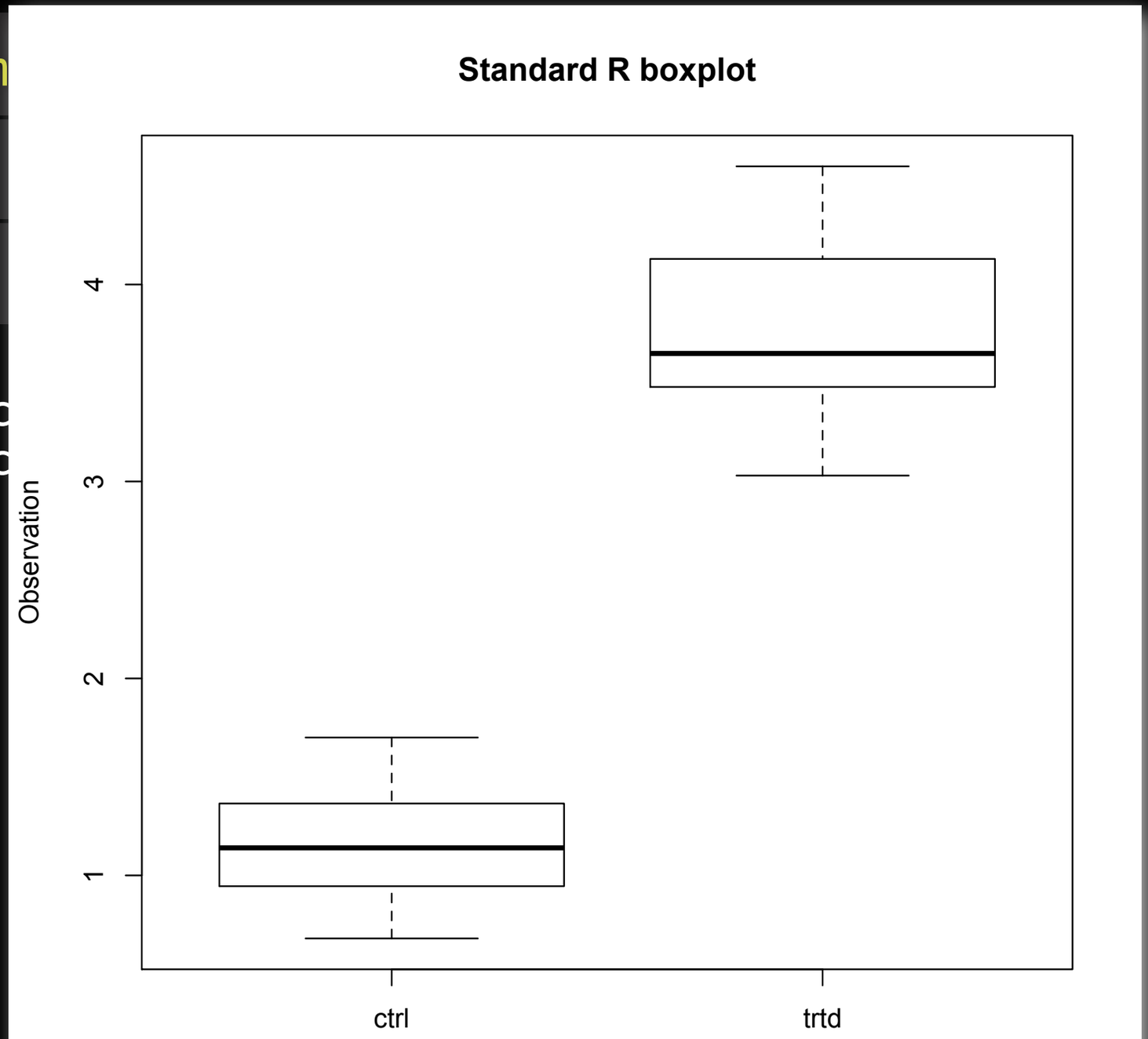

```
R> stripchart(obs ~ grp, data=df,
    method='jitter', xlab="Observation");
```

Always Explore/Visualize Your Data

Group	Sam
∅ control	
treated	

```
R> (df <- data.frame(obs=c
                     grp=c
```

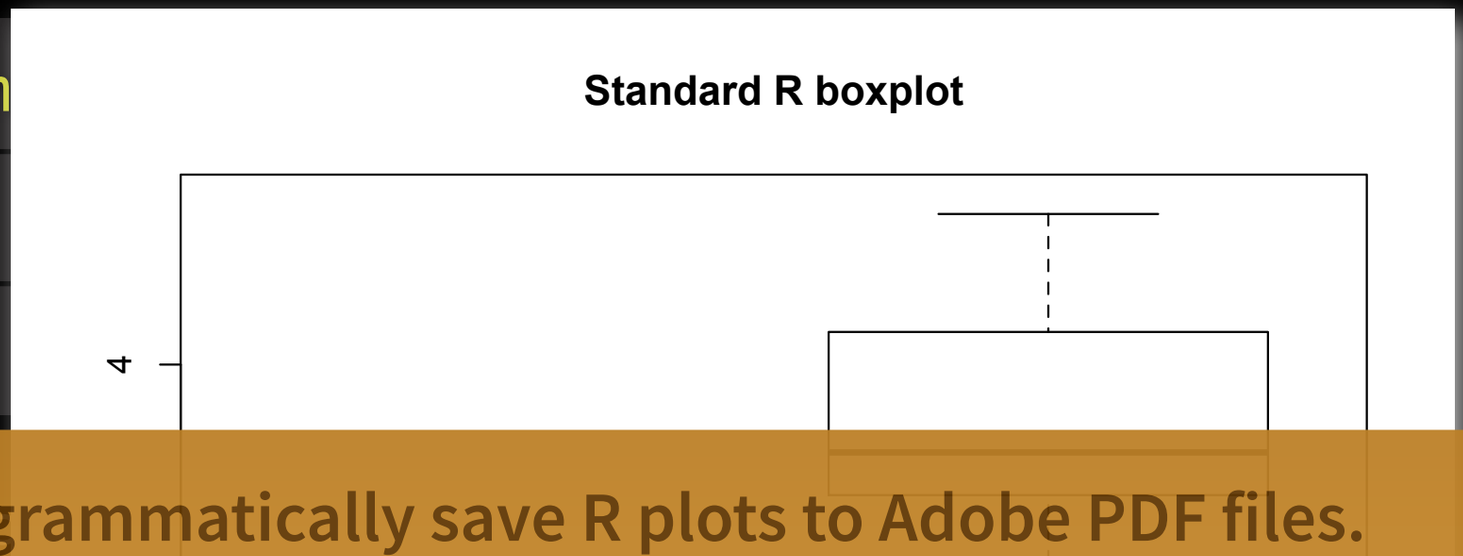
```
    obs  grp
1  1.22 ctrl
2  0.73 ctrl
…… …… ……
19 1.46 ctrl
20 1.27 ctrl
21 4.60 trtd
22 4.39 trtd
…… …… ……
36 3.03 trtd
37 3.88 trtd
```




```
R> boxplot(obs ~ grp, data=df, col='white',
     ylab="Observation", main="Standard R boxplot");
```

Always Explore/Visualize Your Data

Group	Sam
∅ control	
treated	



BTW, you can programmatically save R plots to Adobe PDF files.
This is useful in scripts or, e.g., when running on Hoffman$_2$ when you do not have a graphical display immediately available.

Insert something like this before your plotting command(s):

```
pdf(file="~/Desktop/R-Plot.pdf",
  width=8.5-1-1, height=11-1-1);   # ...sizes are in inches.
```

Make your plot as usual:

```
boxplot(obs ~ grp, data=df, col='white',
  ylab="Observation", main="Standard R boxplot");
```

Follow your plotting command(s) with this:

```
dev.off();
```

```
R> (df <-
    obs   grp
1  1.22  ctrl
2  0.73  ctrl
…… …… ……
19 1.46  ctrl
20 1.27  ctrl
21 4.60  trtd
22 4.39  trtd
…… …… ……
36 3.03  trtd
37 3.88  trtd

R> boxplot(obs ~ grp, data=df, col='white',
     ylab="Observation", main="Standard R boxplot");
```

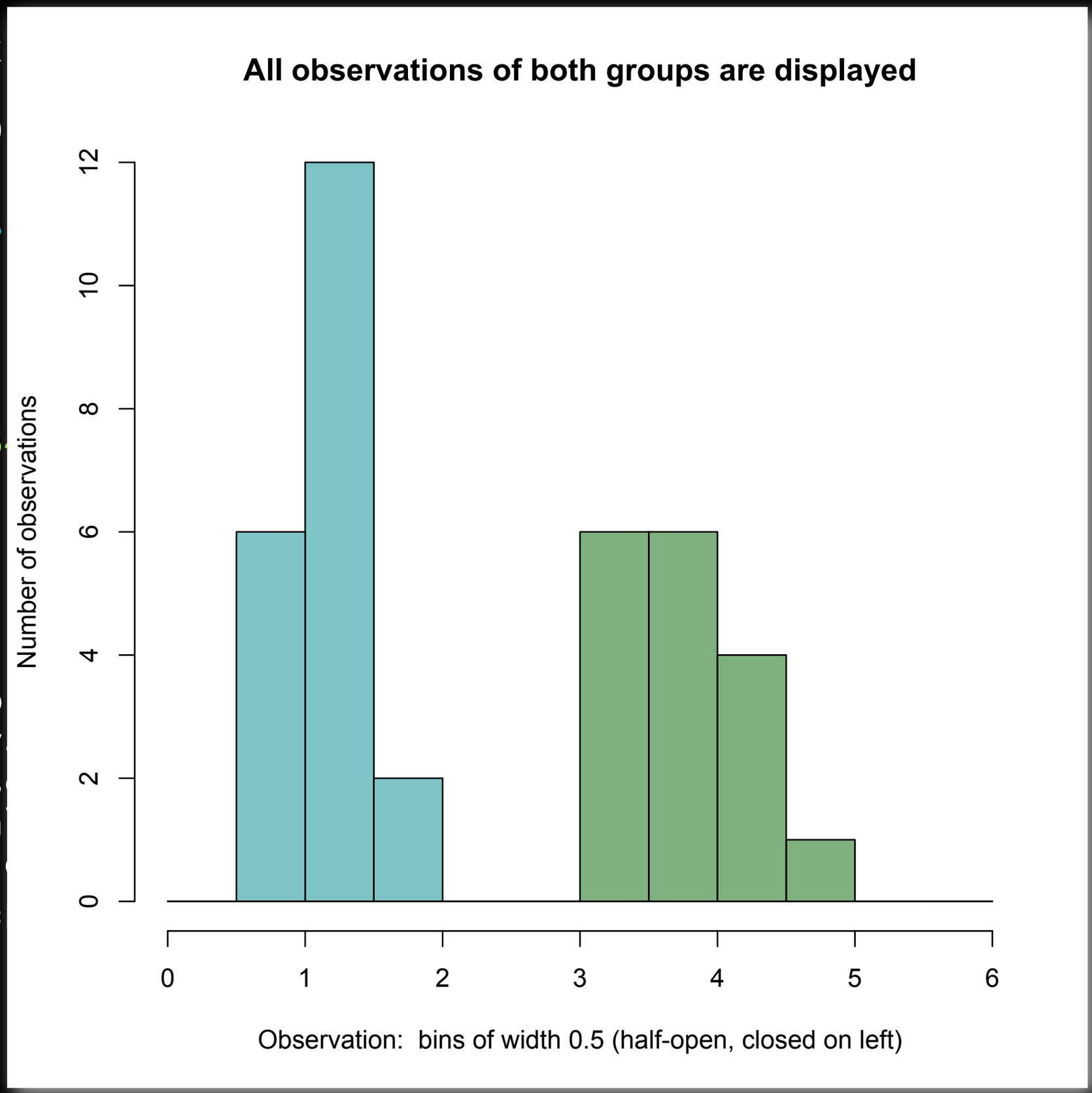
Histograms

```
R> c(min(df$obs), max(df$obs))                  # 0.68 4.60
   range(df$obs)                                # 0.68 4.60
   (brks <- seq(0, 60, 5) / 10)                 # 0.0 0.5 1.0 … 5.5 6.0

      colorC <- "#008B8B80";                    # …hex RGBA



      colorT <- "#00640080";                    # …hex RGBA
```

```
hist(ctrl, breaks=brks, right=FALSE, col=colorC,
    main="All observations of both groups are displayed",
    xlab=paste0("Observation:  bins of width 0.5 ",
                "(half-open, closed on left)"),
    ylab="Number of observations");
 hist(trtd, breaks=brks, right=FALSE, col=colorT, add=TRUE);
```

Histograms

```
R> c(min(df$obs), max
   range(df$obs)
   (brks <- seq(0, 60

      colorC <- "#008

      colorT <- "#006

hist(ctrl, breaks=b
   main="All observa
   xlab=paste0("Obs
               "(ha
   ylab="Number of
 hist(trtd, breaks=
```



Histograms

```
R> c(min(df$obs), max(df$obs))                  # 0.68 4.60
   range(df$obs)                                # 0.68 4.60
   (brks <- seq(0, 60, 5) / 10)                 # 0.0 0.5 1.0 … 5.5 6.0

      colorC <- "#008B8B80";                    # …hex RGBA, or any of:
   # (colorC <- rgb(0,     139,   139, 128, maxColorValue=255))
   # (colorC <- rgb(0,   0.545, 0.545, 0.5))
   # (colorC <- hsv(0.5,   1.0, 0.545, 0.5))

      colorT <- "#00640080";                    # …hex RGBA, or any of:
   # (colorT <- rgb(0,     100,     0, 128, maxColorValue=255))
    # (colorT <- rgb(0,   0.392,     0, 0.5))
    # (colorT <- hsv(1/3,     1, 0.392, 0.5))


hist(ctrl, breaks=brks, right=FALSE, col=colorC,
     main="All observations of both groups are displayed",
     xlab=paste0("Observation:  bins of width 0.5 ",
                 "(half-open, closed on left)"),
     ylab="Number of observations");
 hist(trtd, breaks=brks, right=FALSE, col=colorT, add=TRUE);
```

Histograms

```
R> c(min(df$obs), max(df$obs))                  # 0.68 4.60
   range(df$obs)                                # 0.68 4.60
   (brks <- seq(0, 60, 5) / 10)                 # 0.0 0.5 1.0 … 5.5 6.0

     colorC <- "#008B8B80";                     # …hex RGBA, or any of:
   # (colorC <- rgb(0,     139,   139, 128, maxColorValue=255))
   # (colorC <- rgb(0,   0.545, 0.545, 0.5))
   # (colorC <- hsv(0.5,   1.0, 0.545, 0.5))
   # (colorC <- scales::alpha('darkcyan',  0.5))
     colorT <- "#00640080";                     # …hex RGBA, or any of:
   # (colorT <- rgb(0,     100,     0, 128, maxColorValue=255))
   # (colorT <- rgb(0,   0.392,     0, 0.5))
   # (colorT <- hsv(1/3,     1, 0.392, 0.5))
   # (colorT <- scales::alpha('darkgreen', 0.5))

  hist(ctrl, breaks=brks, right=FALSE, col=colorC,
     main="All observations of both groups are displayed",
     xlab=paste0("Observation:  bins of width 0.5 ",
                 "(half-open, closed on left)"),
     ylab="Number of observations");
   hist(trtd, breaks=brks, right=FALSE, col=colorT, add=TRUE);
```

```
install.packages('scales');
library('scales');
```


```
grDevices::colors()
 # …to see color names

demo('colors')
```

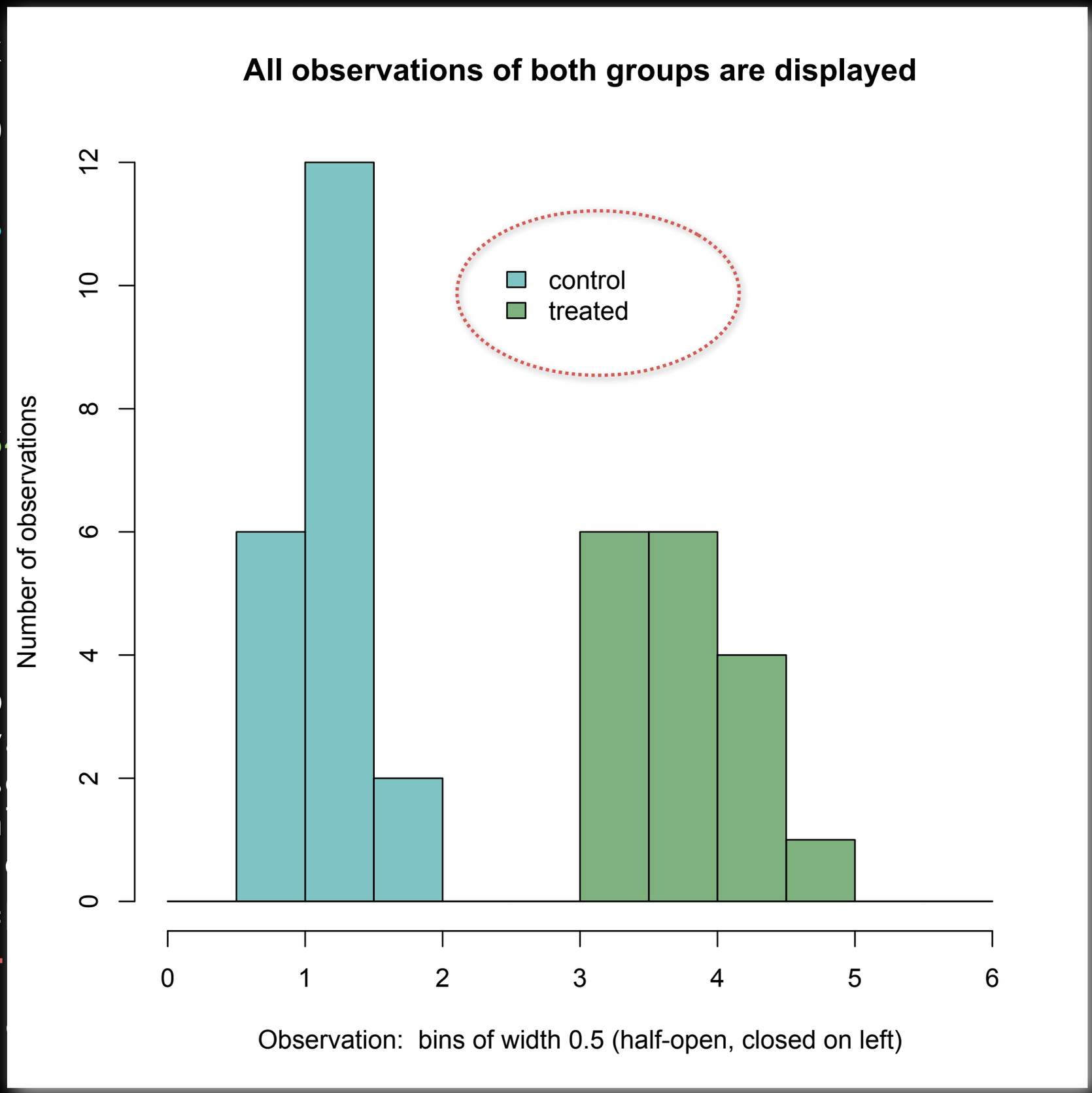
Histograms

```
R> c(min(df$obs), max(df$obs))                  # 0.68 4.60
   range(df$obs)                                # 0.68 4.60
   (brks <- seq(0, 60, 5) / 10)                 # 0.0 0.5 1.0 … 5.5 6.0

      colorC <- "#008B8B80";                    # …hex RGBA



      colorT <- "#00640080";                    # …hex RGBA
```

```
hist(ctrl, breaks=brks, right=FALSE, col=colorC,
   main="All observations of both groups are displayed",
   xlab=paste0("Observation:  bins of width 0.5 ",
               "(half-open, closed on left)"),
   ylab="Number of observations");
 hist(trtd, breaks=brks, right=FALSE, col=colorT, add=TRUE);
 locator(1)  # …INTERACTIVE: WAIT MOUSE CLICK -> (x,y) coords.
 legend(2.3, 10.6, c("control", "treated"),
                   fill=c(colorC, colorT), bty='n');
```

Histograms

```
R> c(min(df$obs), max
   range(df$obs)
   (brks <- seq(0, 60

      colorC <- "#008

      colorT <- "#006

hist(ctrl, breaks=b
   main="All observ
   xlab=paste0("Obs
               "(ha
   ylab="Number of
 hist(trtd, breaks=
 locator(1)  # …INT
 legend(2.3, 10.6,
```

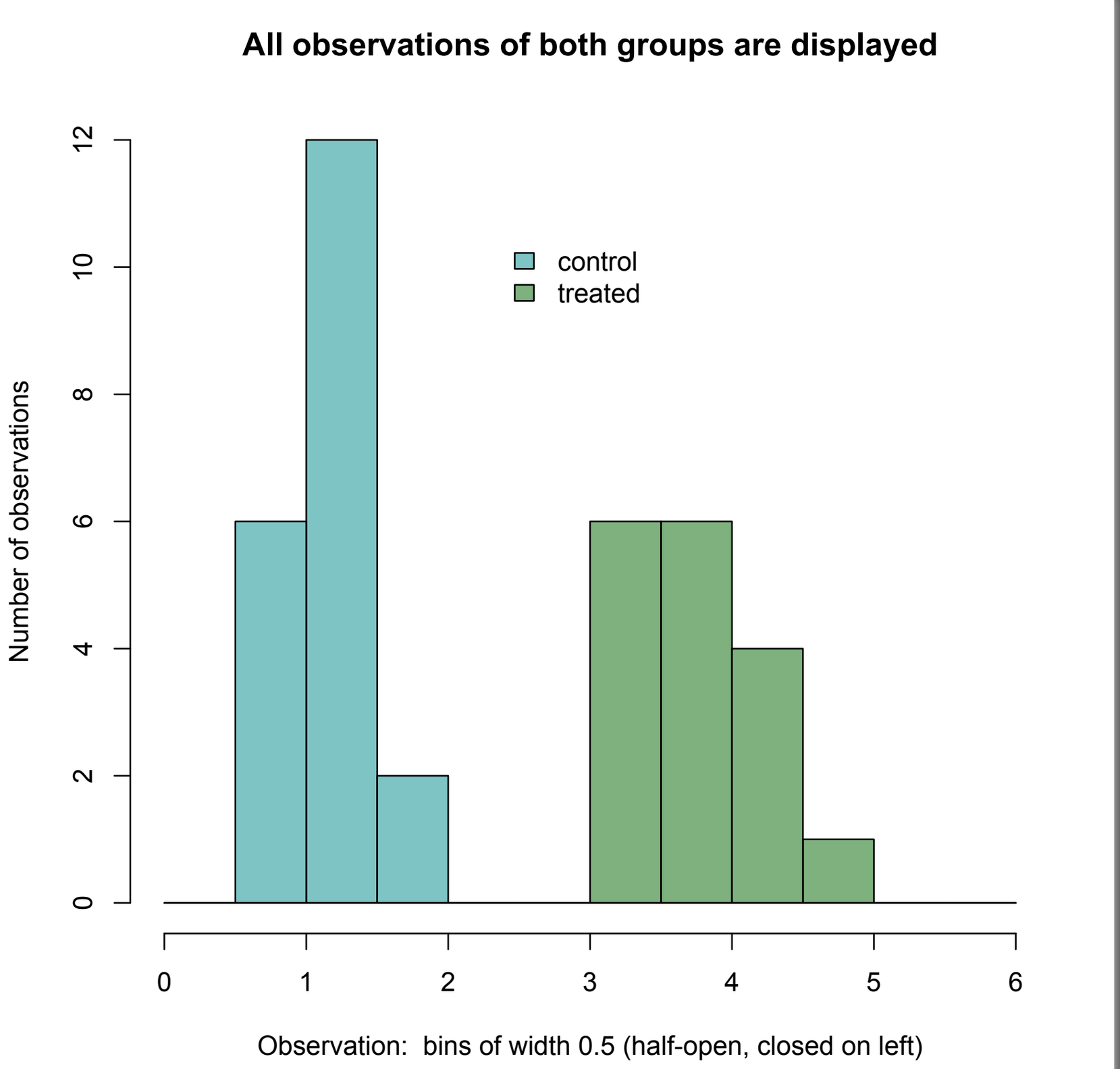


Histograms

CHOOSE HISTOGRAM AXES AND BINS APPROPRIATELY…

- *x* linear or log scale?
- *y* linear or log scale?
- *y* shows counts? probability? density? …non-cumulative or cumulative from left or right?
- avoid bins too small or large
- use same bins when comparing groups; if groups differ greatly in size, consider dual *y* scales
- what happens to points beyond displayed *x* range? — go into first or last bin, or ignored (and if so, were there any?)?
- what happens to points exactly on bin boundaries? — do they go into the bin on left or right? (and not both or neither!)

…AND COMMUNICATE YOUR CHOICES



Discovery within R and the Ecosystem

```
# Help pages:
?c  ?length
?sum  ?mean
?sqrt  ?'<-'
?invisible
?.Last.value
?data.frame
?rep  ?'/'
?stripchart
?boxplot  ?'('
?pdf  ?dev.off
?min  ?max
?range  ?seq
?hist  ?FALSE
?paste0  ?rgb
?hsv  ?'-'
?library  ?'^'
?scales::alpha
?locator  ?'~'
?legend  ?'$'
?install.packages
?'?'
help('$')
help('help')
```

…try looking each new item to you up as you encounter!

```
# Text search installed:
??tukey

# Packages LOADED now
(.packages())

# Package contents?
library(help='base')
library(help='stats')
library(help='graphics')
library(help='grDevices')
library(help='utils')
library(help='datasets')
library(help='scales')
```

…in R Studio, “package:” popup in Environment pane is another way.

```
# All INSTALLED?
library()  # …or R Studio Packages pane.

# Packages loaded by default:
c(getOption('defaultPackages'), 'base')

# GUI/websites can show what
# is AVAILABLE to install
# from CRAN, Bioconductor, …

# “Popularity*” (# downloads):
# CRAN, Bioconductor
#  (*imperfect: includes downloads from
#     dependencies, not just user reqs.)
```

```
# See R implementation?
getAnywhere('alpha')
scales:::alpha
# (for items written in C/C++,
#  Fortran, …, this will not
#  be helpful and you would have
#  to chase down actual sources)


# WE WON’T TALK MUCH ABOUT
# R’s OBJECT SYSTEMS (but
# many packages use them):
# specific implementations
# (“methods”) for “generic”
# (context-aware) functions
# are  «fnBaseName».«class»
print
print.«TAB»
print.data.frame


# Use Google, ChatGPT, …
# to find introductions,
# tutorials, manuals,
# FAQs, popular packages,
# books, documentation,
# etc., especially if you
# have a specific thing
# you want to do or are
# getting into new area!
```

Discovery within R and the Ecosystem

A fair number of R packages bundle decent documentation in themselves. You often find short runnable code examples inside help pages, but especially helpful are medium– or long–form prose "**vignettes**" from package authors as more extensive discussions.

`?mgcv::gam`
[…then, e.g., click "Run examples" in R Studio help browser]

```
vignette()
browseVignettes()

vignette('lmer')
vignette('DESeq2')
RShowDoc('DESeq2', type='html', package='DESeq2')
RShowDoc('survival', type='pdf', package='survival')
```

Note that not all vignettes are available in all forms (HTML, PDF, …).

Of course, these commands only function on packages already installed locally; this is why generally you also need to do Internet searches too.

```
# Help
?c   ?'<-'
?sum
?sqrt
?invisible
?.Last.value
?data
?rep
?stripchart
?boxplot
?pdf
?min
?range
?hist
?paste0   ?rgb
?hsv
?library  ?ALL
?scales::alpha
?locator   ?'~'
?legend   ?'$'
?install.packages

?'?'
help('$')
help('help')

…try looking each
 new item to you up
 as you encounter!
```

```
# Packages loaded by default:
c(getOption('defaultPackages'), 'base')

# GUI/websites can show what
# is AVAILABLE to install
# from CRAN, Bioconductor, …

# "Popularity*" (# downloads):
# CRAN, Bioconductor
#   (*imperfect: includes downloads from
#      dependencies, not just user reqs.)
```

```
# Packages LOADED now
search()
.packages()

# Package contents?
ls()
library(help='base')
library(help='stats')
library(help='graphics')
library(help='grDevices')
library(help='utils')
library(help='datasets')
library(help='methods')
library()  # …or R Studio Packages pane.

scales:::alpha
# (for items written in C/C++,
# Fortran, …, this will not
#  be helpful and you would have
#  to chase down actual sources)

# WE WON'T TALK MUCH ABOUT
# R's OBJECT SYSTEMS (but
# many packages use them):
# specific implementations
# ("methods") of "generic"
# functions
#FALSE
# e.g. «fnBaseName».«class»
print
print.«TAB»
print.data.frame
```

```
# Use Google, ChatGPT, …
# to find introductions,
# tutorials, manuals,
# FAQs, popular packages,
# books, documentation,
# etc., especially if you
# have a specific thing
# you want to do or are
# getting into new area!
```

Discovery within R and the Ecosystem

```
# Help pages:
?c  ?length
?sum  ?mean
?sqrt  ?'<-'
?invisible
?.Last.value
?data.frame
?rep  ?'/'
?stripchart
?boxplot  ?'('
?pdf  ?dev.off
?min  ?max
?range  ?seq
?hist  ?FALSE
?paste0  ?rgb
?hsv  ?'-'
?library  ?'^'
?scales::alpha
?locator  ?'~'
?legend  ?'$'
?install.packages
?'?'
help('$')
help('help')
```

…try looking each new item to you up as you encounter!

BTW, everything in R pretty much boils down to (sometimes special) **function calls, and R allows more diverse characters in symbols/names than most languages. There's quite a bit of "syntactic sugar"** (which makes the language more pleasant and accessible).

```
a <- c('p','q','r');
3 + 4                    is  `+`(3, 4)
a[2]                     is  `[`(a, 2)
c(2,3) %o% c(5,7)        is  `%o%`(c(2,3), c(5,7))
a[2:3] <- c('s','t')     is  a <- `[<-`(a, 2:3,
                                 values=c('s','t'))
```

Things like `data.frame` are just names that happen to have a period in them.

```
# GUI/websites can show what
# is AVAILABLE to install
# from CRAN, Bioconductor, …
# "Popularity*" (# downloads):
# CRAN, Bioconductor
#   (*imperfect: includes downloads from
#      dependencies, not just user reqs.)
```

```
# WE WON'T TALK MUCH ABOUT
# BIG OBJECT SYSTEMS (but
# many packages use them):
# specific implementations
# ("methods") for "generic"
# functions
# are «fnBaseName».«Class»
# print.TBD
print.data.frame
# Use Google, ChatGPT, …
# to find introductions,
# tutorials, manuals,
# FAQs, popular packages,
# books, documentation,
# etc., especially if you
# have a specific thing
# you want to do or are
# getting into new area!
```

Interrupting Running R

```
R> for(i in 1:100) {
     hist(runif(1000, min=10, max=30),
       breaks=9:31, ylim=c(0,80), main=i);
     Sys.sleep(0.2);
     }
```

THINGS TO TRY IF R IS BUSY AND YOU WANT IT TO STOP:

«ESCAPE» key or type «COMMAND-.» (⌘-period),
or click GUI "STOP sign" button

If you are running R from a UNIX command line then typing, e.g., «CTRL-C» or «CTRL-Z» may help

Note that sometimes R is busy inside non-R code (e.g., C/C++/Fortran/…) that has not always been written to have interruptible points. In some cases, **you may have to force quit R** — ***make sure you are saving your R scripts and/or R workspaces often so you do not lose much work then!***

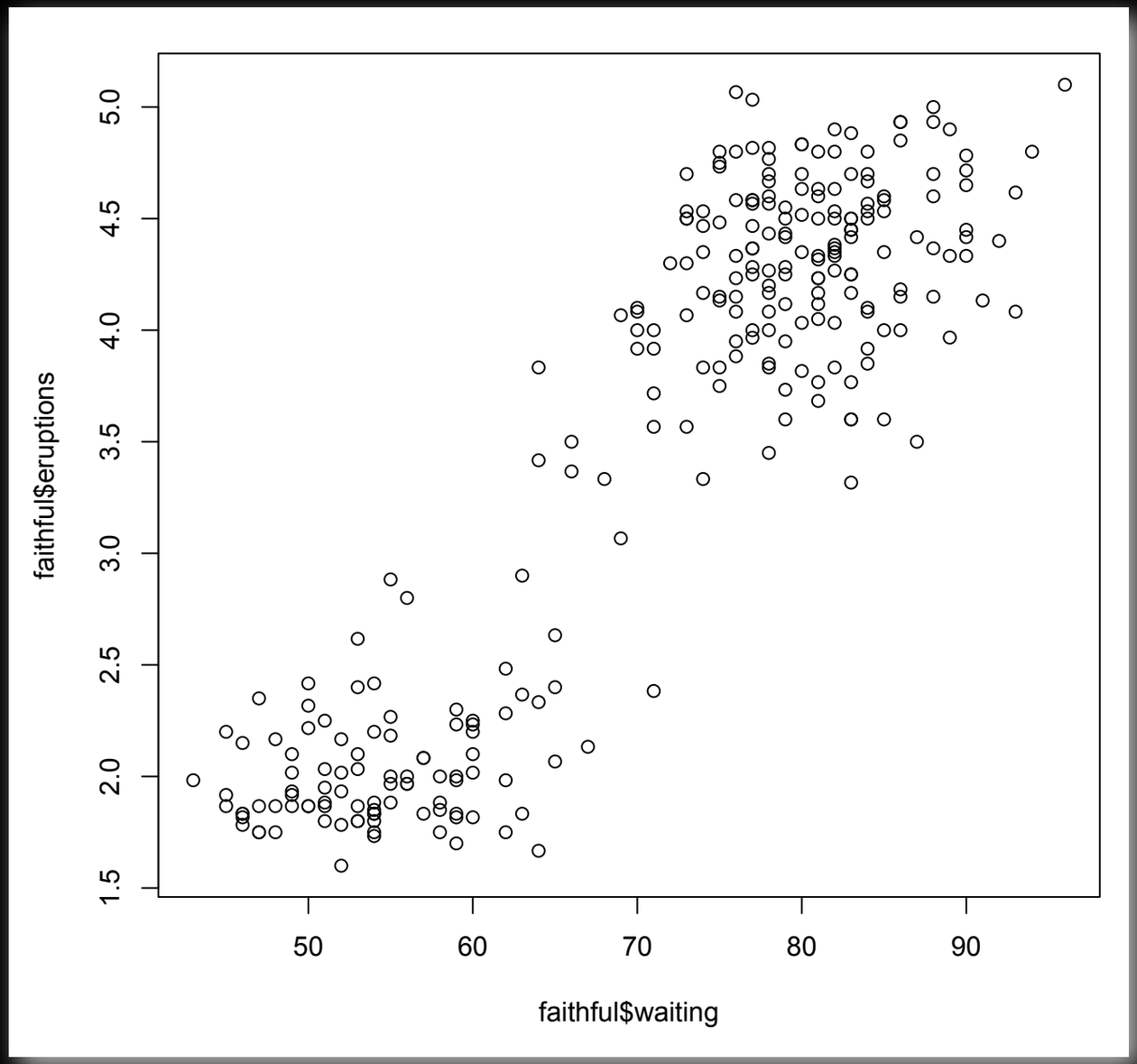
In general, you can get out–of–control resource usage of different types: not just CPU time, but also RAM memory or disk space… any of these may require a force quit to recover (also possible the operating system kills it by itself).

Very important is the discipline to keep your scripts/notes in good order so that running your notes from a fresh R reproduces your analysis.

Peek Forward: Smoothed Histograms?

```
faithful     # …nice small built in dataset:  Old Faithful timings
?faithful    # …brief documentation on the dataset
# BASIC INSPECTION TO GET AN IDEA OF WHAT THIS DATA IS LIKE:

plot(faithful$waiting, faithful$eruptions)  # …lacks jitter, and
faithful[ faithful$eruptions == 1.75, ]     #  there are dupe pts:
```

```
    eruptions waiting
14       1.75      47
……       ……        ……
22       1.75      47
……       ……        ……
……       ……        ……
```



Peek Forward: Smoothed Histograms?

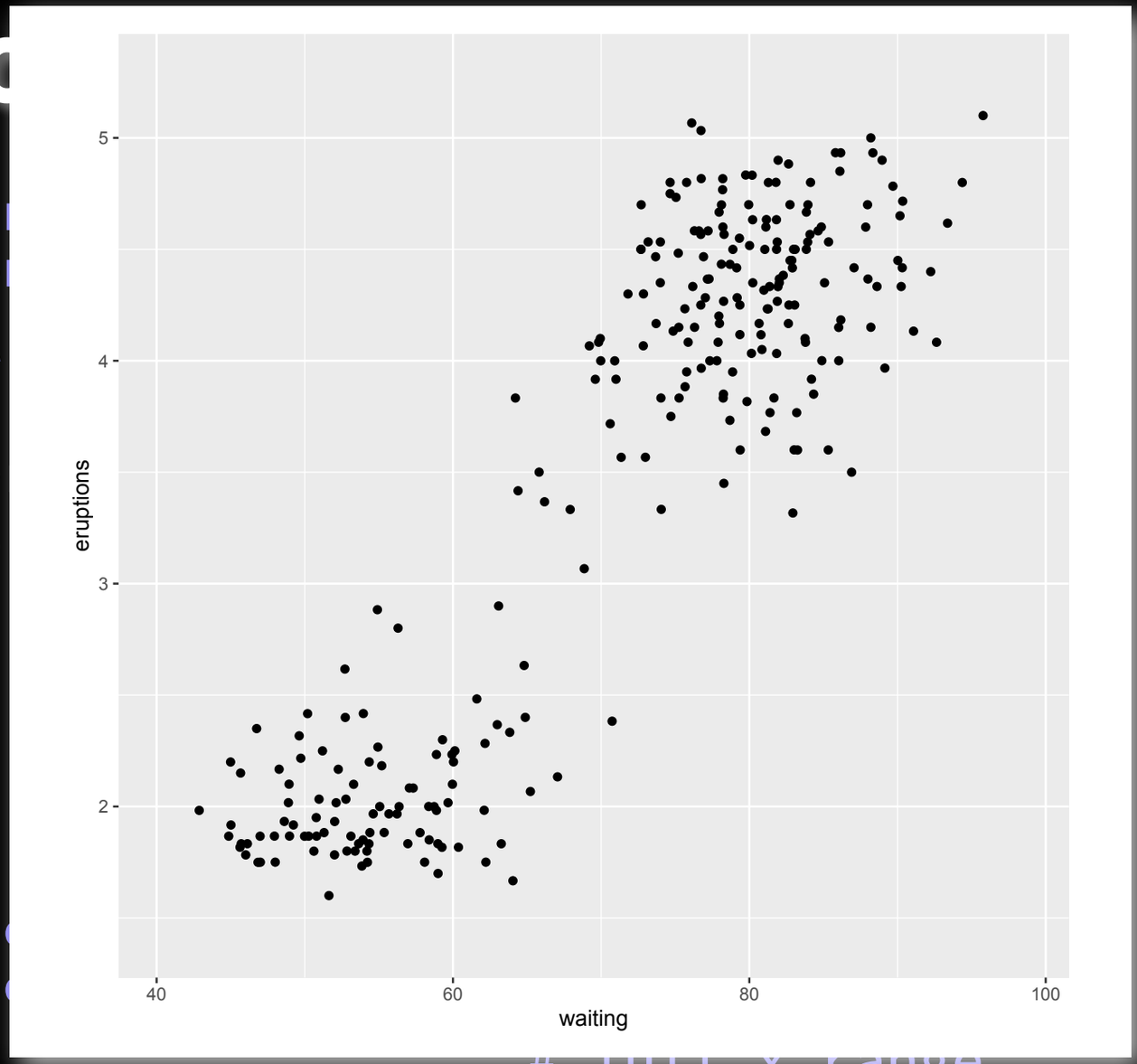
```
faithful     # …nice small built in dataset:  Old Faithful timings
?faithful    # …brief documentation on the dataset
# BASIC INSPECTION TO GET AN IDEA OF WHAT THIS DATA IS LIKE:

plot(faithful$waiting, faithful$eruptions)  # …lacks jitter, and
faithful[ faithful$eruptions == 1.75, ]     #  there are dupe pts:
```

```
    eruptions waiting
14       1.75      47
……       ……        ……
22       1.75      47
……       ……        ……
```


```
library('ggplot2');   # if you need:  install.packages('ggplot2')
library('ggExtra');   # if you need:  install.packages('ggExtra')
(xLim <- range(faithful$waiting))                    # full x range
(yLim <- range(faithful$eruptions))                  # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))  # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))  # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
          scale_x_continuous(limit=xLim)                +
          scale_y_continuous(limit=yLim)                +
          geom_point(position='jitter'))  # …run this a few times
```

Peek Forward: Smo

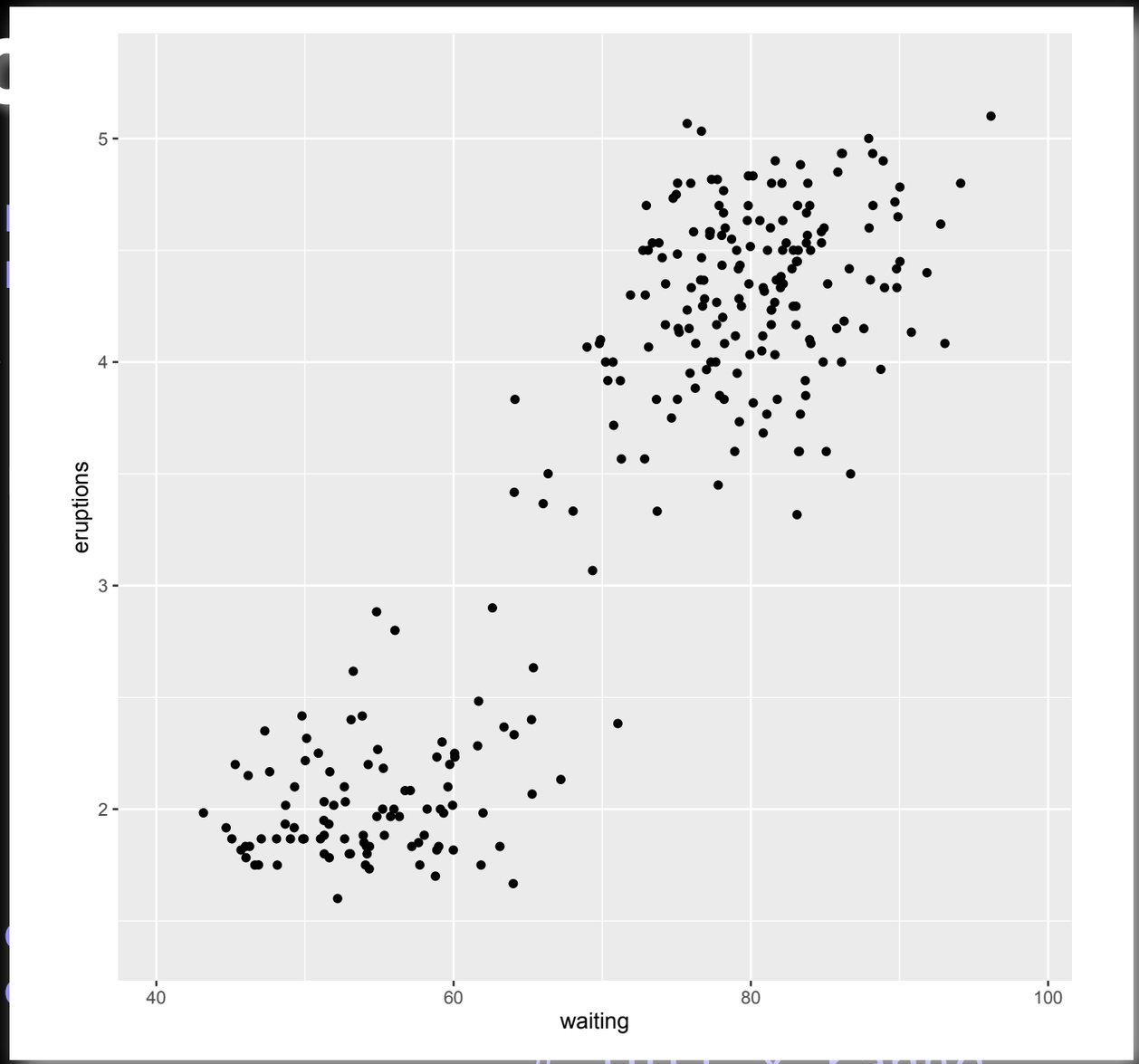
```
faithful     # …nice small built i
?faithful    # …brief documentatio
# BASIC INSPECTION TO GET AN IDEA

plot(faithful$waiting, faithful$e
faithful[ faithful$eruptions == 1
    eruptions waiting
14       1.75      47
……       ……        ……
22       1.75      47
……       ……        ……

library('ggplot2');    # if you ne
library('ggExtra');    # if you ne
(xLim <- range(faithful$waiting))                    # full x range
(yLim <- range(faithful$eruptions))                  # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))  # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))  # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
          scale_x_continuous(limit=xLim)                +
          scale_y_continuous(limit=yLim)                +
          geom_point(position='jitter'))  # …run this a few times
```



Peek Forward: Smo

```
faithful     # …nice small built i
?faithful    # …brief documentatio
# BASIC INSPECTION TO GET AN IDEA

plot(faithful$waiting, faithful$e
faithful[ faithful$eruptions == 1
```

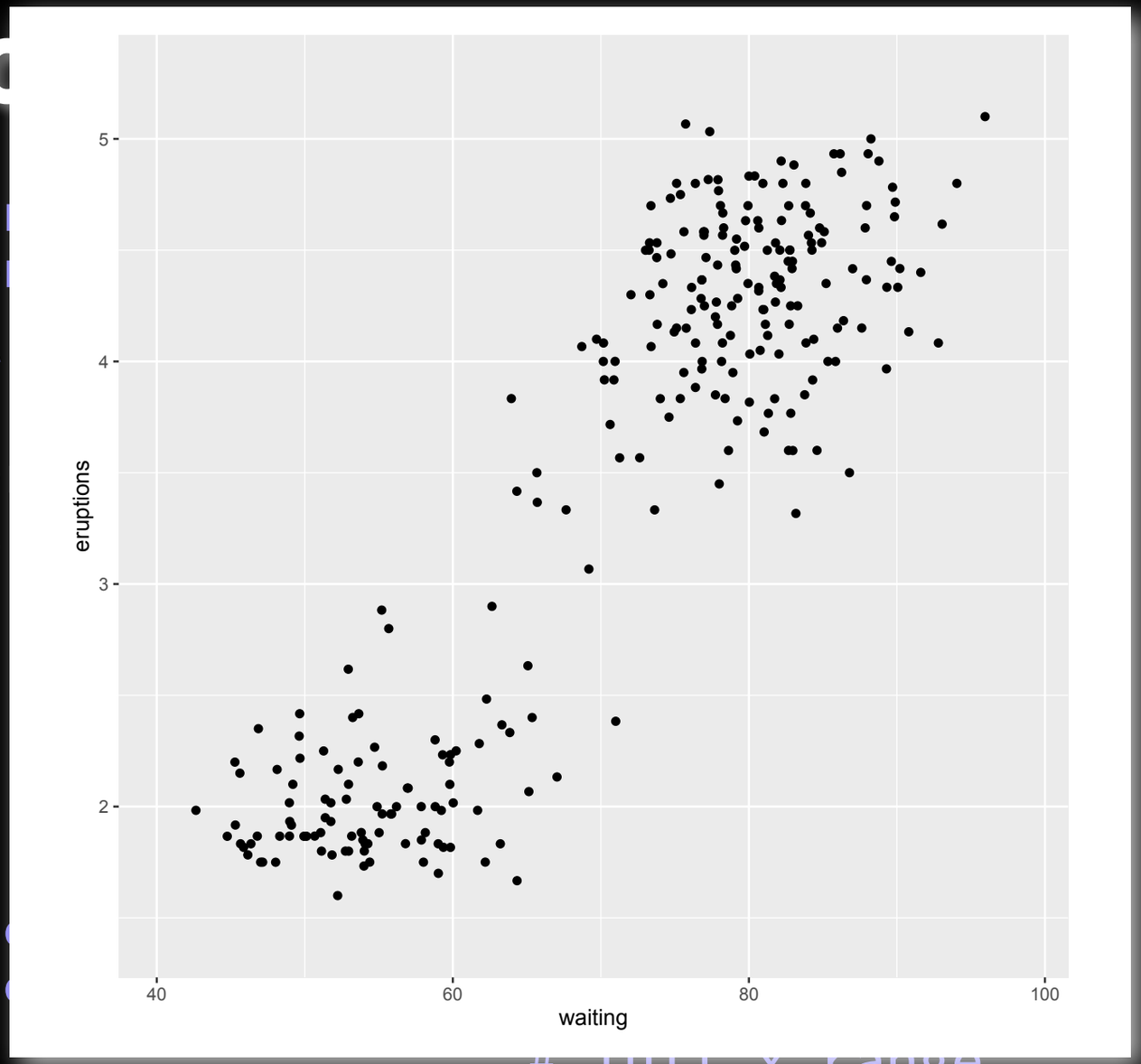
	eruptions	waiting
14	1.75	47
.....	.....	.....
22	1.75	47
.....	.....	.....




```
library('ggplot2');    # if you ne
library('ggExtra');    # if you ne
(xLim <- range(faithful$waiting))                          # full x range
(yLim <- range(faithful$eruptions))                        # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))  # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))  # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
           scale_x_continuous(limit=xLim)                  +
           scale_y_continuous(limit=yLim)                  +
           geom_point(position='jitter'))  # …run this a few times
```

Peek Forward: Smo

```
faithful     # …nice small built i
?faithful    # …brief documentatio
# BASIC INSPECTION TO GET AN IDEA

plot(faithful$waiting, faithful$e
faithful[ faithful$eruptions == 1
    eruptions waiting
14        1.75      47
……        ……        ……
22        1.75      47
……        ……        ……

library('ggplot2');    # if you ne
library('ggExtra');    # if you ne
(xLim <- range(faithful$waiting))                    # full x range
(yLim <- range(faithful$eruptions))                  # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))  # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))  # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
           scale_x_continuous(limit=xLim)                +
           scale_y_continuous(limit=yLim)                +
           geom_point(position='jitter'))  # …run this a few times
```



Peek Forward: Smoothed Histograms?

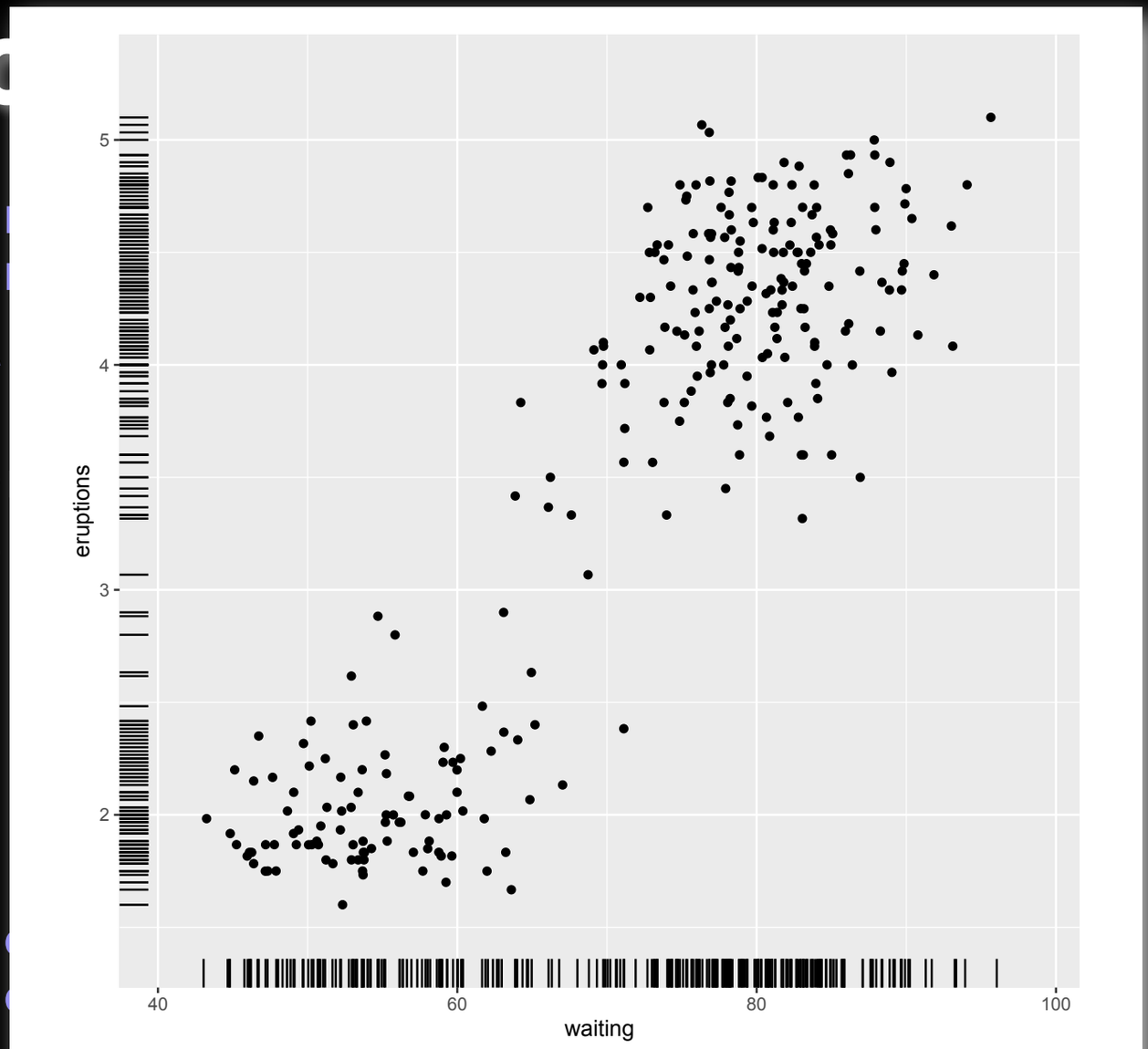
```
faithful     # …nice small built in dataset:  Old Faithful timings
?faithful    # …brief documentation on the dataset
# BASIC INSPECTION TO GET AN IDEA OF WHAT THIS DATA IS LIKE:

plot(faithful$waiting, faithful$eruptions)  # …lacks jitter, and
faithful[ faithful$eruptions == 1.75, ]     #  there are dupe pts:
```

```
    eruptions waiting
14       1.75      47
……       ……        ……
22       1.75      47
……       ……        ……
```

```
library('ggplot2');   # if you need:  install.packages('ggplot2')
library('ggExtra');   # if you need:  install.packages('ggExtra')
(xLim <- range(faithful$waiting))                    # full x range
(yLim <- range(faithful$eruptions))                  # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))  # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))  # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
           scale_x_continuous(limit=xLim)              +
           scale_y_continuous(limit=yLim)              +
           geom_point(position='jitter'))  # …run this a few times
plt1 + geom_rug(position='jitter')         # …run this a few times
```

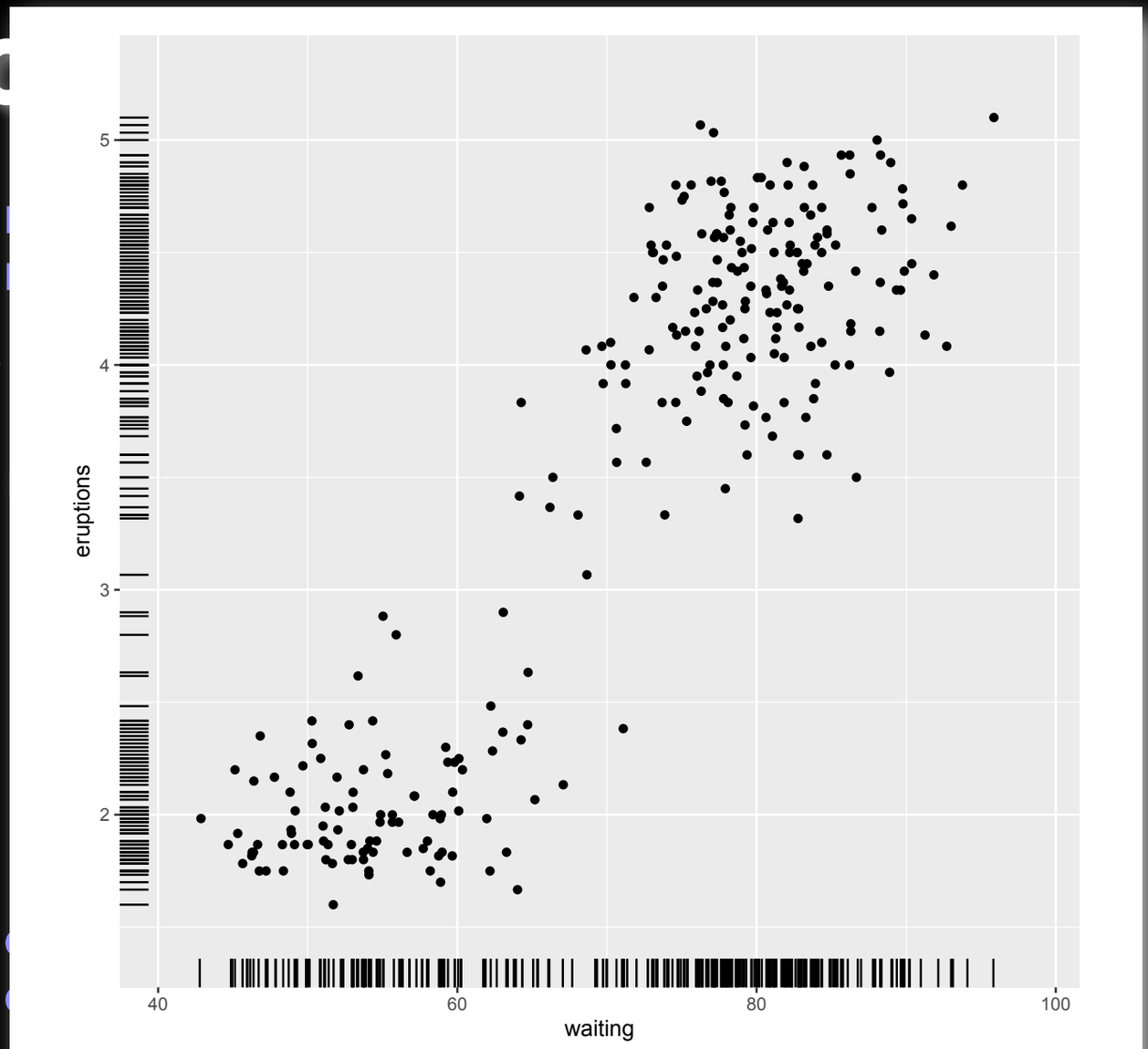
Peek Forward: Smo

```
faithful     # …nice small built i
?faithful    # …brief documentatio
# BASIC INSPECTION TO GET AN IDEA

plot(faithful$waiting, faithful$e
faithful[ faithful$eruptions == 1
    eruptions waiting
14        1.75      47
……        ……        ……
22        1.75      47
……        ……        ……

library('ggplot2');    # if you ne
library('ggExtra');    # if you ne
(xLim <- range(faithful$waiting))                    # full x range
(yLim <- range(faithful$eruptions))                  # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))      # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))      # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
           scale_x_continuous(limit=xLim)            +
           scale_y_continuous(limit=yLim)            +
           geom_point(position='jitter'))  # …run this a few times
plt1 + geom_rug(position='jitter')         # …run this a few times
```



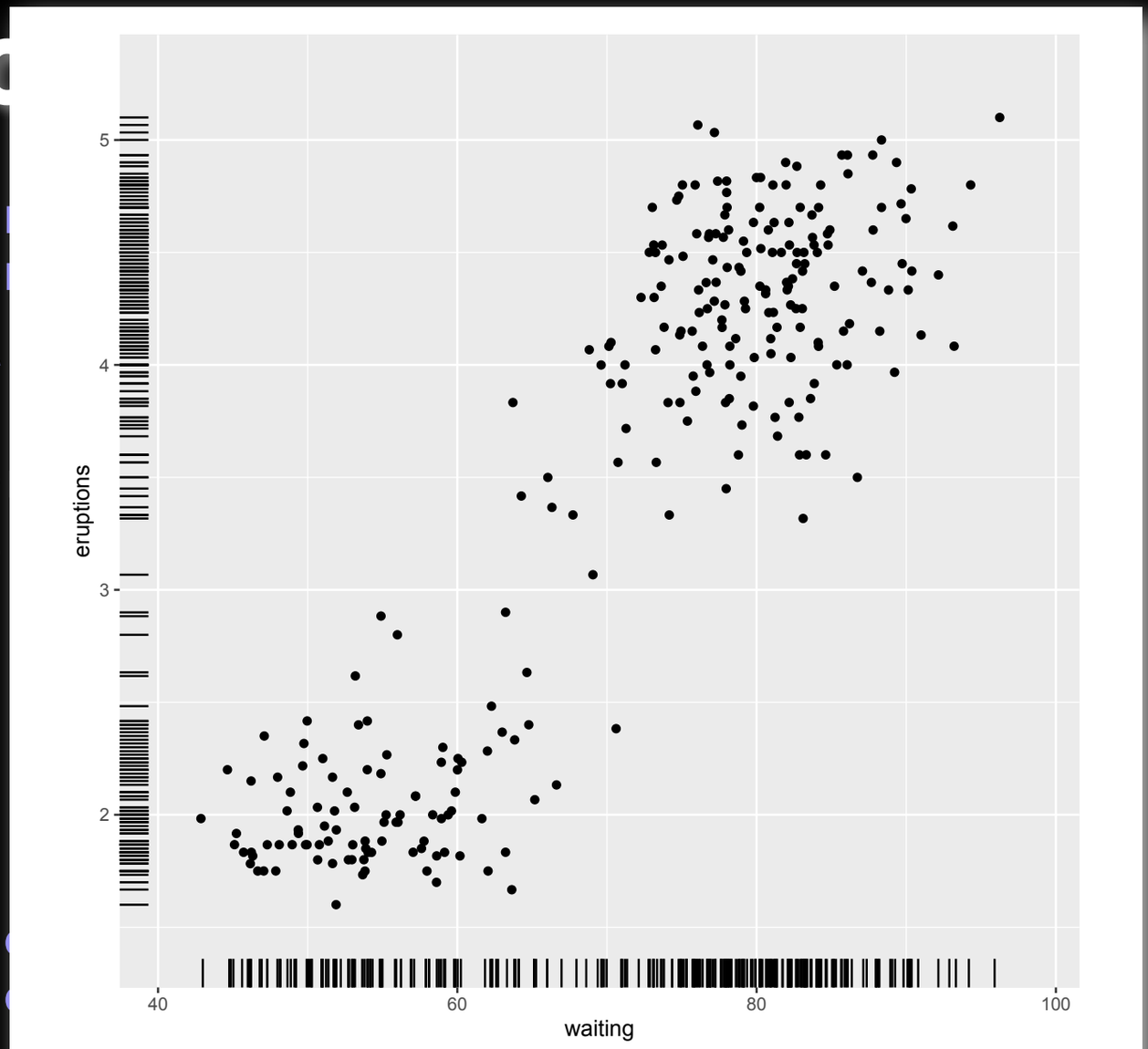
Peek Forward: Smo

```
faithful     # …nice small built i
?faithful    # …brief documentatio
# BASIC INSPECTION TO GET AN IDEA

plot(faithful$waiting, faithful$e
faithful[ faithful$eruptions == 1
    eruptions waiting
14       1.75      47
……       ……        ……
22       1.75      47
……       ……        ……

library('ggplot2');   # if you ne
library('ggExtra');   # if you ne
(xLim <- range(faithful$waiting))                    # full x range
(yLim <- range(faithful$eruptions))                  # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))  # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))  # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
           scale_x_continuous(limit=xLim)                +
           scale_y_continuous(limit=yLim)                +
           geom_point(position='jitter'))  # …run this a few times
plt1 + geom_rug(position='jitter')         # …run this a few times
```



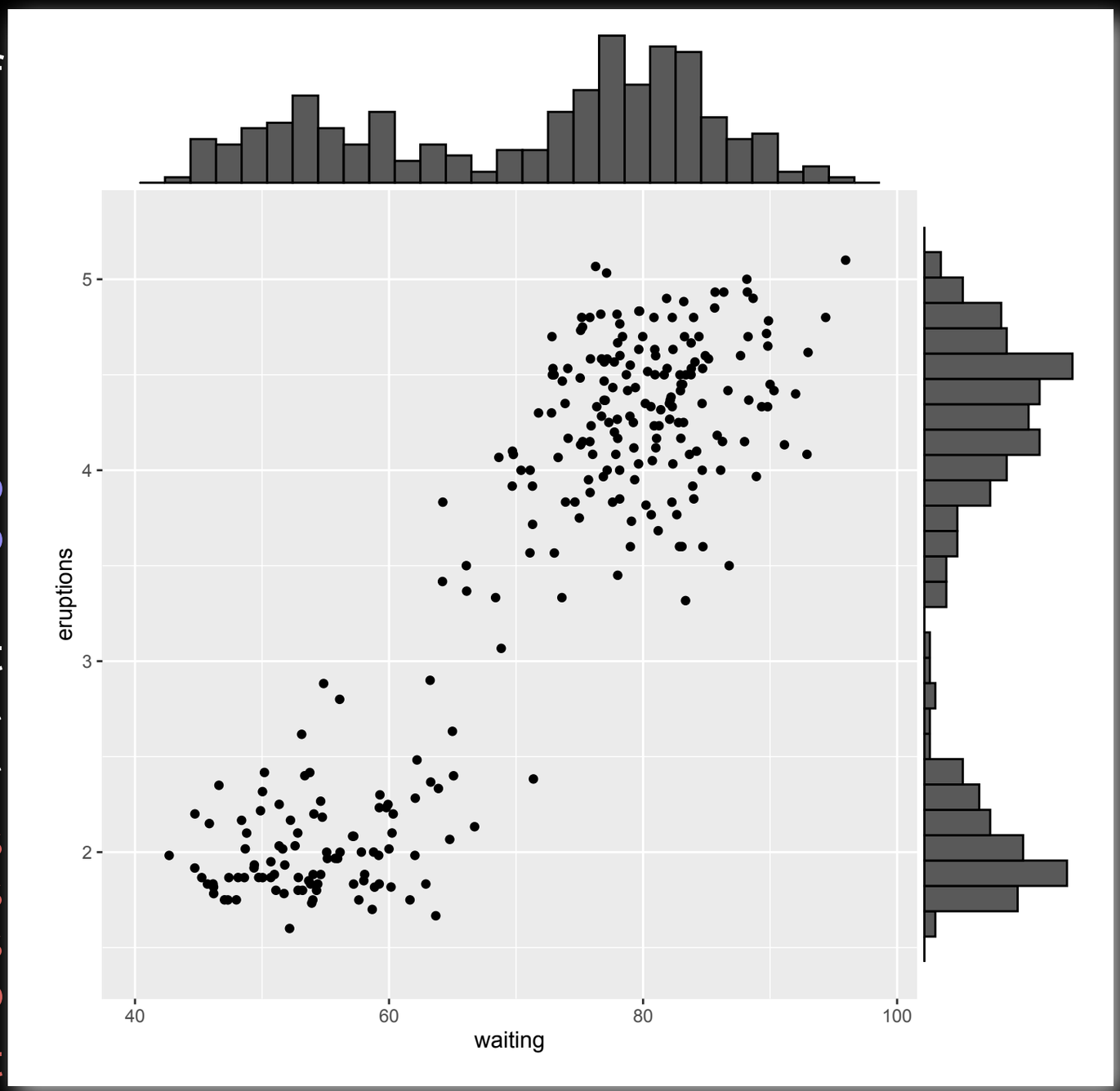
Peek Forward: Smo

```
faithful     # …nice small built i
?faithful    # …brief documentatio
# BASIC INSPECTION TO GET AN IDEA

plot(faithful$waiting, faithful$e
faithful[ faithful$eruptions == 1
    eruptions waiting
14       1.75      47
……       ……        ……
22       1.75      47
……       ……        ……

library('ggplot2');    # if you ne
library('ggExtra');    # if you ne
(xLim <- range(faithful$waiting))                    # full x range
(yLim <- range(faithful$eruptions))                  # full y range
(xLim <- xLim + 0.05*c(-1,1)*(xLim[2]-xLim[1]))      # x: go ±5% more
(yLim <- yLim + 0.05*c(-1,1)*(yLim[2]-yLim[1]))      # y: go ±5% more
(plt1 <- ggplot(faithful, aes(x=waiting, y=eruptions)) +
           scale_x_continuous(limit=xLim)            +
           scale_y_continuous(limit=yLim)            +
           geom_point(position='jitter'))  # …run this a few times
plt1 + geom_rug(position='jitter')         # …run this a few times
```



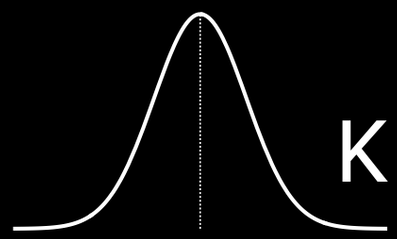
Peek Forward: Smoothed Histograms?

```
faithful     # …nice small built in dataset:  Old Faithful timings
?faithful    # …brief documentation on the dataset
# BASIC INSPECTION TO GET AN IDEA OF WHAT THIS DATA IS LIKE:

plot(faithful$waiting, faithf
faithful[ faithful$eruptions
    eruptions waiting
14        1.75      47
.....     .....     .....
22        1.75      47
.....     .....     .....

library('ggplot2');    # if yo
library('ggExtra');    # if yo
(xLim <- range(faithful$waiti
(yLim <- range(faithful$erupt
(xLim <- xLim + 0.05*c(-1,1)*
(yLim <- yLim + 0.05*c(-1,1)*
(plt1 <- ggplot(faithful, aes
          scale_x_continuous
          scale_y_continuous
          geom_point(positio
plt1 + geom_rug(position='jit
ggMarginal(plt1, type='histogram')   # …many params. at defaults.
```



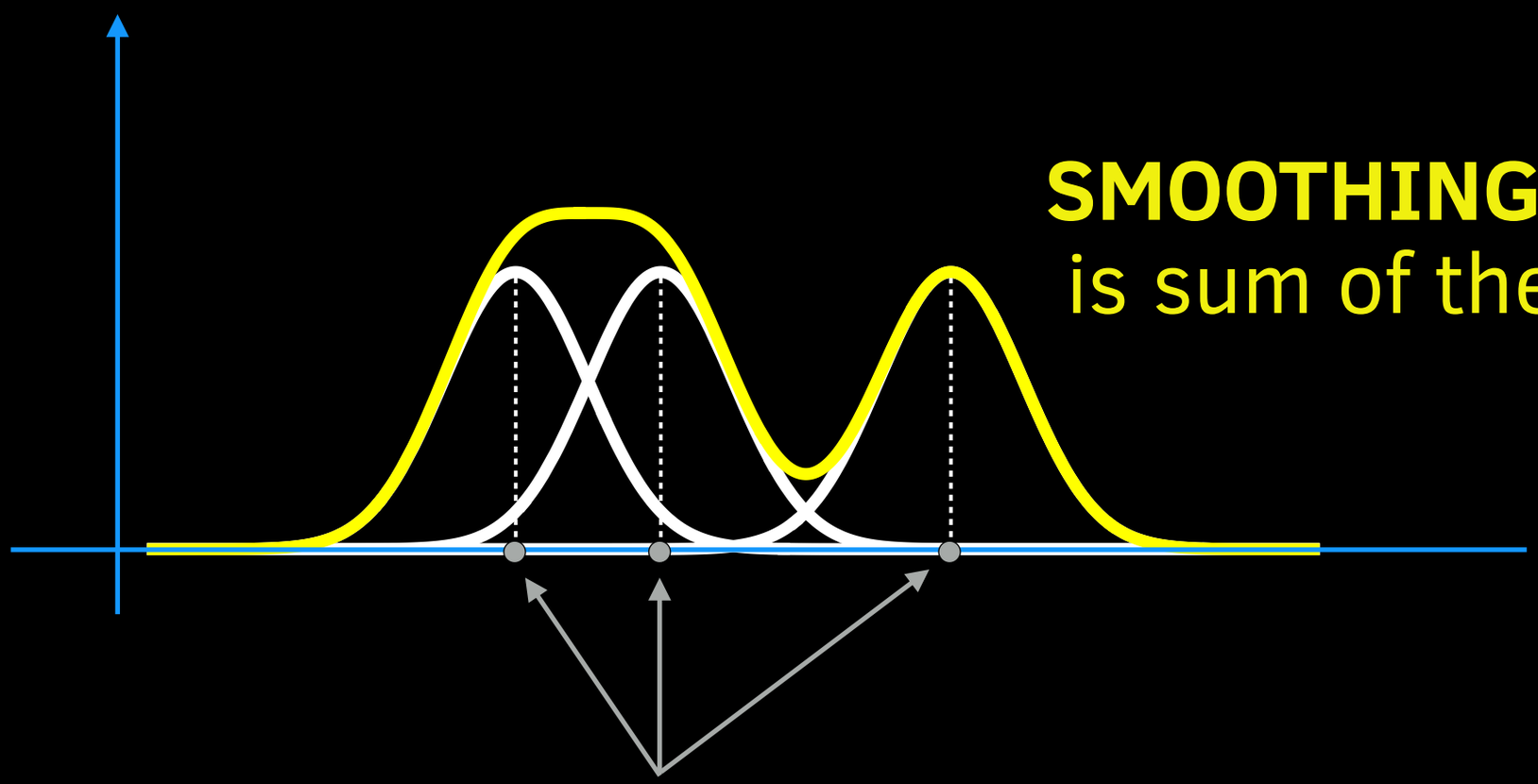
Kernel Smoothing: 1–D case

(e.g., for density estimation or visualization)



KERNEL FUNCTION

(many choices; “width” is “bandwidth”)

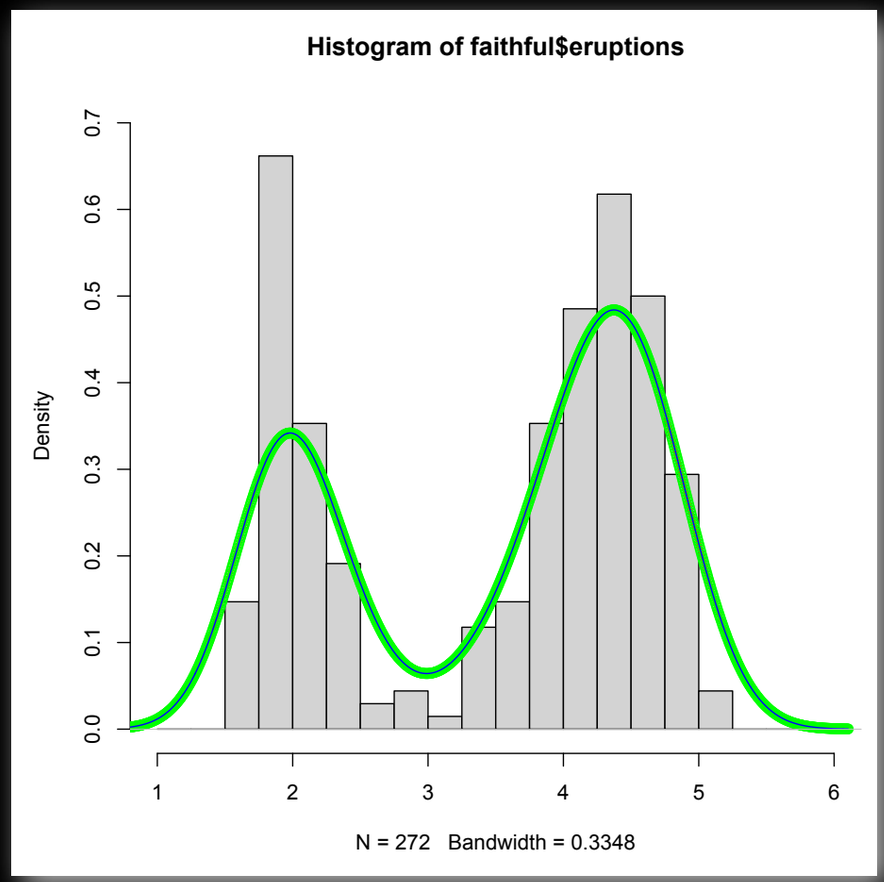


SMOOTHING (up to *y*-scaling)
is sum of the kernel copies

DATA POINTS:
put a copy of kernel
above each one

Quick Visualize Faithful Data This Way

```
xLim <- c(1,6);  yLim <- c(0,0.7);
dens <- density(faithful$eruptions, bw='nrd0', adjust=1.0,
          kernel='gaussian', window='gaussian', n=1024, cut=3.0);
hist(faithful$eruptions, breaks=seq(1,6,0.25),
    freq=FALSE, xlim=xLim, ylim=yLim, xlab='');
  lines(dens$x, dens$y, col='yellow4', lwd=4);  # <= plot option 1
  par(new=TRUE);                                #
  plot(dens$x, dens$y, xlim=xLim, ylim=yLim,    # <= plot option 2
    col='green', axes=FALSE, main='', ylab='', xlab='');
  par(new=TRUE);                                #
  plot(dens, xlim=xLim, ylim=yLim, col='blue',  # <= plot option 3
       axes=FALSE, main='', ylab='');           #
```




```
# Numerous R functions/packages have
#  this kind of thing built in, such as:
ggMarginal(plt1, type='density')
ggplot(faithful, aes(x=eruptions))                +
  geom_histogram(aes(y=after_stat(density))) +
  geom_density(color='red')
# …and there are 2-D, etc. generalizations
```

How R Values are Put Together

ALMOST ALL THE VALUES WE CARE ABOUT IN R ARE EITHER…

LIST: finite sequence of ≥0 *potentially arbitrary* R values (“pairlists” are a detail we can ignore)

or ATOMIC **VECTOR:** finite sequence of ≥0 *definitely homogeneous* R values… (we ignore “raw” vectors)

- logical: TRUE, FALSE + missing value (NA) (avoid using `T` and `F`)
- integer: signed 32-bit twos complement + missing value (`NA_integer_`)
  representable range is `-2147483648L .. 2147483647L`
- numeric: IEEE 754 double precision floating point (“binary64”) real numbers, including missing value (`NA_real_`), signed infinities (`±Inf`), signed zeros (well hidden), denormals, not-a-numbers (`NaN`); largest contiguous integer range is ±9,007,199,254,740,992, normals in binary: 1.«52 mantissa bits» • 2^{-1022} to 2^{+1023}, 15–17 decimal digits, normals ≈ $\pm 10^{\pm 308}$ (denorms. ≈ $\pm 5 \cdot 10^{-324}$)

  R tends to auto-upgrade/downgrade between integers and numerics, and you can usually consider integers and numerics interchangeable
- complex: real part numeric + imaginary part numeric (e.g., `2.5-3i`), and missing value (`NA_complex_`)
- character: strings of len. ≥0 (ASCII, UNICODE UTF-8, …) + `NA_character_`

R *(like MATLAB)* ***has no scalars!*** *…instead it just uses vectors of length 1*

WE DON’T HAVE TO CARE VERY MUCH ABOUT OTHER KINDS (NULL, object system objects, language-relateds [symbols, environments, calls/formulae, expressions], function-relateds [closures, promises, specials, builtins, bytecode, formal argument-related], or more exotics [external pointers, weak references]), or various internal flags (debug, trace, …).

TWIST: *EVERY* R VALUE HAS “**ATTRIBUTES**” … a sequence of ≥0 key (strings/symbols)–arbitrary value pairs… some keys are magic (`names`, `dim`, `dimnames`, `class`, `tsp`, `row.names`, …).

How R Values are Put Together

ALMOST ALL THE VALUES WE CARE ABOUT IN R ARE EITHER...

LIST: `(L <- list(nameIdx1="value1", 2.0, `rare!`=TRUE))`

or ATOMIC **VECTOR:**

logical: `(vL <- c(TRUE, FALSE, third=TRUE, NA, FALSE))`
integer: `(vI <- c(-3L, secondName=4L, NA_integer_))`

numeric: `(vN <- c(3.4, name2=1.5e-10, -Inf, NaN))`

complex: `(vC <- c(1, a=3.0, a=2-4i, 1/(4+7i), NA))`

character: `(vS <- c(front="HEAD", "middle", back="Tail"))`

R has no scalars: `length(3.4) == 1` `3.4[[1]] == 3.4` `3.4[1] == 3.4`

```
c(class(L), class(vL), class(vI), class(vN), class(vC), class(vS))
[1] "list"      "logical"   "integer"   "numeric"   "complex"   "character"

attributes(L)                                # $names: [1] "nameIdx1" "" "name3"
attr(L, 'names', exact=TRUE)                 # [1] "nameIdx1" "" "name3" (…a character vector)
attr(L, 'names') <- c('aaa','b','!')         # L elt. names now  "aaa" "b" "!"
```

How R Values are Taken Apart

BOTH LISTS AND VECTORS CAN BE TAKEN APART IN SEVERAL WAYS...

```
(L  <- list(aaa="value1", b=2.0, `!`=TRUE))       # list
(vL <- c(TRUE, FALSE, third=TRUE, NA, FALSE))     # logical   vector
(vI <- c(-3L, secondName=4L, NA_integer_))        # integer   vector
(vN <- c(3.4, name2=1.5e-10, -Inf, NaN))          # numeric   vector
(vC <- c(1, a=3.0, a=2-4i, 1/(4+7i), NA))         # complex   vector
(vS <- c(front="HEAD", "middle", back="Tail"))    # character vector
```

EXTRACT SINGLE ELEMENT, removing any name attached to the element:

```
L[[1]]       vN[[2]]          ...by 1-based position
L[["aaa"]]   vN[["name2"]]    ...by names/dimnames attributes
L$aaa                         ...by names/dimnames attributes     FOR LISTS ONLY
L$a                              Allows unique prefix partial name matches!
                                 First match for repeated names?
L$`!`                         ...backquotes protect "weird" names
L[["NO"]]    L$NO             ...missing names give NULL          FOR LISTS ONLY
```

EXTRACT POTENTIALLY MULTIPLE ELEMENTS ("SLICE"); result has same type as started with:

```
4:7                              ...evaluates to integer vector c(4L, 5L, 6L, 7L)
L[c(2,3)]   vN[c(2,3)]           ...list/numeric — keeping names — with 2nd & 3rd elt.
L[2:3]      vN[2:3]              ...same as L[c(2,3)] and vN[c(2,3)]
L[-2]       vN[-2]               ...integer index -n means all-but-index-n
vN[c(2,0,0,2,2,1)]               ...elts. 2nd repeated 3x, then 1st;  0's are ignored
vS[c(FALSE,FALSE,TRUE)]          ...elts. in the TRUE positions;  how stuff like v[v<2]<-0 works
L[c("b", "!")]   L[]             ...by names/dimnames attributes;  omitted = keep everything
```

PRETTY MUCH ALL FORMS CAN BE USED ON *BOTH* SIDES OF ASSIGNMENTS: LHS <- RHS, ...

See, e.g., `?'[['` help page for additional gory details, and sometimes you just have to empirically experiment.
SOME VALUES ARE MULTI-DIMENSIONAL, USE MORE THAN ONE INDEX: *matrix*[*row*, *col*] *dataFrame*[*rowCase*, *colVar*] ...

How R Values are Taken Apart

Because R "scalars" are really vectors of length 1, this leads to the **common confusion of many R users between `vec[[1]]` and `vec[1]` indexing**, because for vectors without named elements these will behave the same!

```
vec <- c(9,2,7,4);
vec[[1]]
vec[1]
```

This is also why you get all those "`[1]` …" leaders in front of many printed values: lines start with "`[`index`]`" of lead element

EXTRACT SINGLE ELEMENT, removing any name attached to the element:

`L[[1]]`	`vN[[2]]`	…by 1-based position
`L[["aaa"]]`	`vN[["name2"]]`	…by `names`/`dimnames` attributes
`L$aaa`		…by `names`/`dimnames` attributes **FOR LISTS ONLY**
`L$a`		Allows unique prefix partial name matches!
		First match for repeated names?
`L$`!``		…backquotes protect "weird" names
`L[["NO"]]`	`L$NO`	…missing names give `NULL` **FOR LISTS ONLY**

EXTRACT POTENTIALLY MULTIPLE ELEMENTS ("SLICE"); result has same type as started with:

`4:7`		…evaluates to integer vector `c(4L, 5L, 6L, 7L)`
`L[c(2,3)]`	`vN[c(2,3)]`	…list/numeric — keeping names — with 2nd & 3rd elt.
`L[2:3]`	`vN[2:3]`	…same as `L[c(2,3)]` and `vN[c(2,3)]`
`L[-2]`	`vN[-2]`	…integer index *−n* means all-but-index-*n*
`vN[c(2,0,0,2,2,1)]`		…elts. 2nd repeated 3x, then 1st; `0`'s are ignored
`vS[c(FALSE,FALSE,TRUE)]`		…elts. in the `TRUE` positions; how stuff like `v[v<2]<-0` works
`L[c("b", "!")]`	`L[]`	…by `names`/`dimnames` attributes; omitted = keep everything

PRETTY MUCH ALL FORMS CAN BE USED ON *BOTH* SIDES OF ASSIGNMENTS: `LHS <- RHS, …`

See, e.g., `?'[['` help page for additional gory details, and sometimes you just have to empirically experiment.
SOME VALUES ARE MULTI-DIMENSIONAL, USE MORE THAN ONE INDEX: *matrix*[*row*, *col*] *dataFrame*[*rowCase*, *colVar*] …

Inspect a Possibly Complicated R Value

```
dens <- density(faithful$eruptions, bw='nrd0', adjust=1.0,
          kernel='gaussian', window='gaussian', n=1024, cut=3.0);
```

Option 1: expand entry in R Studio ‘Environment’ pane; sometimes click to pop into a further detail pane and expand more there. (…not always complete)

Option 2: use core R functionality…

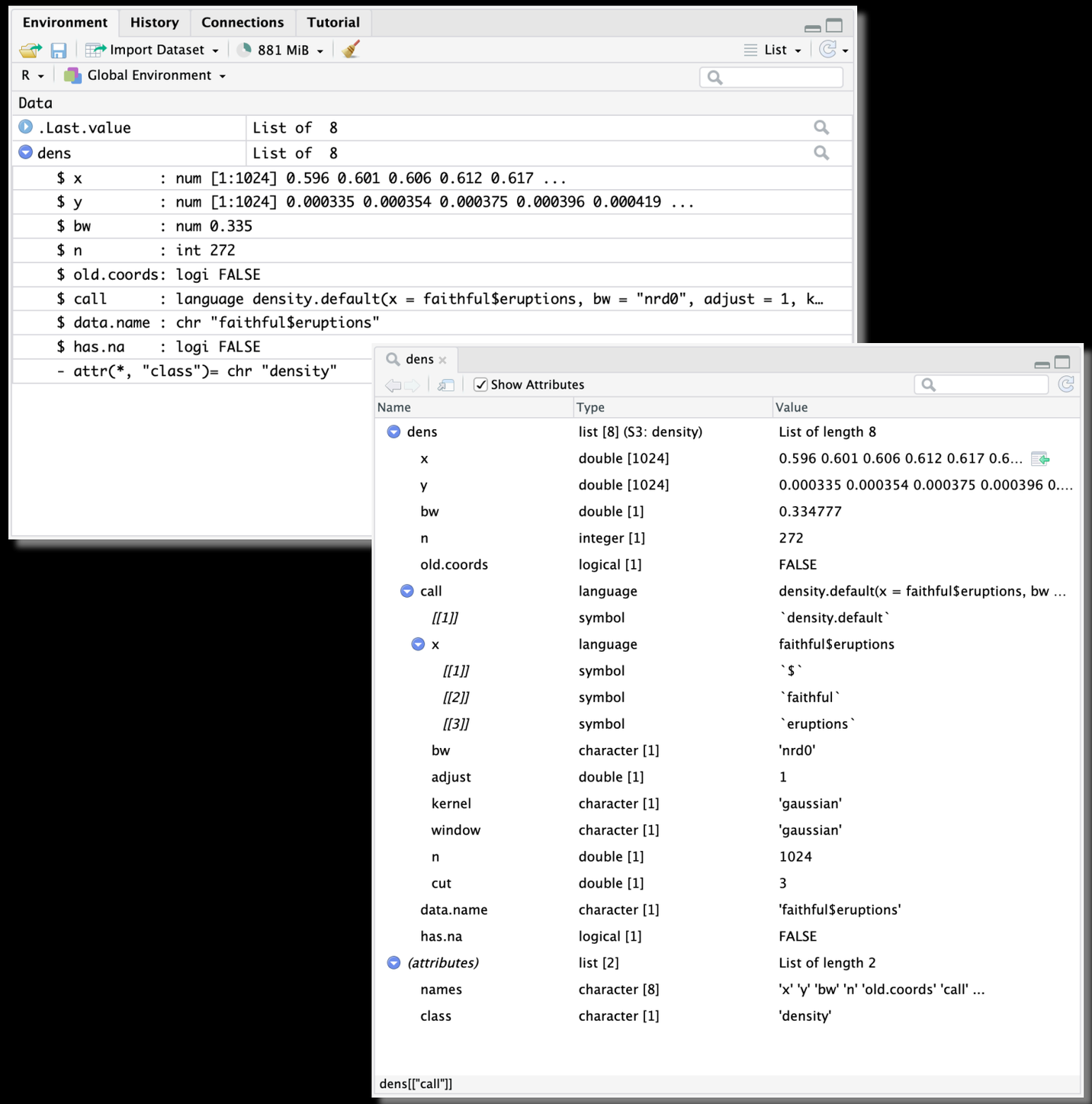
```
typeof(value)
mode(value)
storage.mode(value)
class(value)
attributes(value)
length(value)
```

…and recurse into all elements, using `value[[index]]` **or** `value$name`**, and recurse into all attributes, using** `attr(value, "attributeName")`

Option 3: use R debug features…

```
str(value)
dump('varName', file='')
```

Also, TAB completion in GUI can help: type `dens$` and then hit «TAB» key



Inspect a Possibly Complicated R Value

```
dens <- density(faithful$eruptions, bw='nrd0', adjust=1.0,
          kernel='gaussian', window='gaussian', n=1024, cut=3.0);
```

Option 1: expand entry in R Studio ‘Environment’ pane; sometimes click to pop into a further detail pane and expand more there. (…not always complete)

Option 2: use core R functionality…

```
typeof(value)
mode(value)
storage.mode(value)
class(value)
attributes(value)
length(value)
```

…and recurse into all elements, using *value*`[[`*index*`]]` **or** *value*`$`*name*, **and recurse into all attributes, using** `attr(`*value*`, "`*attributeName*`")`

Option 3: use R debug features…

```
str(value)
dump('varName', file='')
```

Also, TAB completion in GUI can help: type `dens$` and then hit «TAB» key

```
typeof(dens)          # [1] "list"
mode(dens)            # [1] "list"
storage.mode(dens)    # [1] "list"
class(dens)           # [1] "density"
attributes(dens)
$names
[1] "x"         "y"           "bw"
[4] "n"         "old.coords"  "call"
[7] "data.name" "has.na"

$class
[1] "density"
length(dens)          # [1] 8
```

Inspect a Possibly Complicated R Value

```
dens <- density(faithful$eruptions, bw='nrd0', adjust=1.0,
          kernel='gaussian', window='gaussian', n=1024, cut=3.0);
```

Option 1: expand entry in R Studio ‘Environment’ pane; sometimes click to pop into a further detail pane and expand more there. (…not always complete)

Option 2: use core R functionality…

```
typeof(value)
mode(value)
storage.mode(value)
class(value)
attributes(value)
length(value)
```

…and recurse into all elements, using `value[[index]]` **or** `value$name`**, and recurse into all attributes, using** `attr(value, "attributeName")`

Option 3: use R debug features…

```
str(value)
dump('varName', file='')
```

Also, TAB completion in GUI can help:
type `dens$` and then hit «TAB» key

```
typeof(dens$x)          # [1] "double"
mode(dens$x)            # [1] "numeric"
storage.mode(dens$x)    # [1] "double"
class(dens$x)           # [1] "numeric"
attributes(dens$x)      # NULL
length(dens$x)          # [1] 1024
dens$x                  # …
plot(dens$x)            # …
```

«… and repeat for other 7 elements …»

« also repeat for: `attr(dens, "names")` »

« also repeat for: `attr(dens, "class")` »

If some of these elements are themselves compound objects or have attributes of their own, then would recurse on those.

Inspect a Possibly Complicated R Value

```
dens <- density(faithful$eruptions, bw='nrd0', adjust=1.0,
          kernel='gaussian', window='gaussian', n=1024, cut=3.0);
```

Option 1: expand entry in R Studio ‘Environment’ pane; sometimes click to pop into a further detail pane and expand more there. (…not always complete)

Option 2: use core R functionality…

```
typeof(value)
mode(value)
storage.mode(value)
class(value)
attributes(value)
length(value)
```

…and recurse into all elements, using *value*`[[`*index*`]]` **or** *value*`$`*name*, **and recurse into all attributes, using** `attr(`*value*`, "`*attributeName*`")`

Option 3: use R debug features…

```
str(value)
dump('varName', file='')
```

Also, TAB completion in GUI can help: type `dens$` and then hit «TAB» key

```
str(dens)
```

```
List of 8
 $ x         : num [1:1024] 0.596 0.601 0.606 0.612 0.617 ...
 $ y         : num [1:1024] 0.000335 0.000354 0.000375 0.000396 0.000419 ...
 $ bw        : num 0.335
 $ n         : int 272
 $ old.coords: logi FALSE
 $ call      : language density.default(x = faithful$eruptions, bw = "nrd0",
                         adjust = 1, kernel = "gaussian",
                         window = "gaussian", n| __truncated__
 $ data.name : chr "faithful$eruptions"
 $ has.na    : logi FALSE
 - attr(*, "class")= chr "density"
```

```
# str(…)  tries to give you a complete
# overview, but abbreviate long data.
```

```
dump('dens', file='')
```

```
dens <-
structure(list(x = c(0.59566889660817113, 0.6010537081495041, …
6.0935614803091624, 6.0989462918504955, 6.1043311033918286),
    y = c(0.00033520798755312192, 0.00035428794866818893, …,
    0.00022273634499121964), bw = 0.33477703446394302, n = 272L,
    old.coords = FALSE, call = quote(density.default(x = faithful$eruptions,
        bw = "nrd0", adjust = 1, kernel = "gaussian", window = "gaussian",
        n = 1024, cut = 3)), data.name = "faithful$eruptions",
    has.na = FALSE), class = "density")
```

```
# dump(…)  tries to give you R code
# that would reconstruct the value.
```

Inspect a Possibly Complicated R Value

```
dens <- density(faithful$eruptions, bw='nrd0', adjust=1.0,
          kernel='gaussian', window='gaussian', n=1024, cut=3.0);
```

Option 1: expand entry in R Studio 'Environment' pane; sometimes click to pop into a further detail pane and expand more there. (…not always complete)

Option 2: use core R functionality…

```
typeof(value)
mode(value)
storage.mode(value)
class(value)
attributes(value)
length(value)
```

…and recurse into all elements, using `value[[index]]` **or** `value$name`**, and recurse into all attributes, using** `attr(value, "attributeName")`

Option 3: use R debug features…

```
str(value)
dump('varName', file='')
```

Also, TAB completion in GUI can help: type `dens$` and then hit «TAB» key

Now you see how to discover that `dens$x` and `dens$y` have the smoothed points, and also that there is metadata available such as `dens$bw` and `dens$n`. In this case, help page `?density` does have a *Value* section where these are documented, but everything you want is not always so.

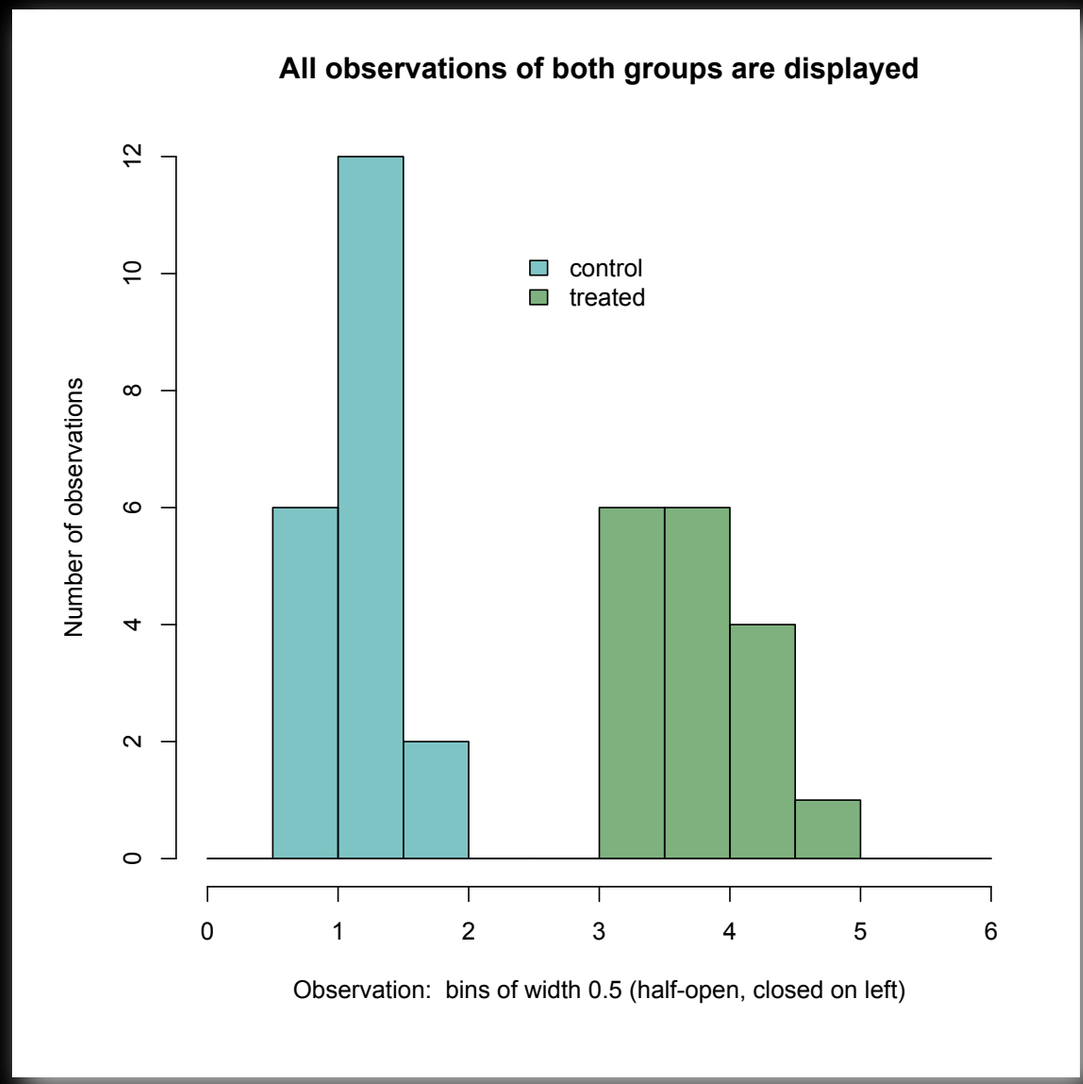
Sometimes packages provide accessor functions that are preferred to use to extract information from package outputs (in case internal details change), but these may not be complete, so you have to dig in yourself.

Further, to understand how some package-assisted analysis is working, it greatly helps to know what data/results are going into/out of each stage of the analysis. Direct inspection of R values and objects makes that possible.

Recall Our Simple Running Example...


```
R> (df <- data.frame(obs=c(ctrl, trtd),
                     grp=c(rep("ctrl", length(ctrl)),
                           rep("trtd", length(trtd)) )))

    obs  grp
1  1.22 ctrl
2  0.73 ctrl
..... ..... .....
19 1.46 ctrl
20 1.27 ctrl
21 4.60 trtd
22 4.39 trtd
..... ..... .....
36 3.03 trtd
37 3.88 trtd
```

BTW, data.frame's use *row.names* and *names* attributes for row and column names, resp.



Practicing Discovery/Inspection/Use

```
# Although this is sort-of jumping ahead, if you have had some
# statistics before, perhaps your reaction to  df  was to think
# “linear models / regression / ANOVA”.  So maybe we know the
# word “ANOVA” for a technique we are interested in, but don’t
# know any ways to get R to do it
```

observed responses (df$obs) = MODEL$_{modelParameters}$(predicting variates (df$grp)) + residuals

Some universe of possible models described by particular parameters

MODEL FITTING optimizes the modelParameters to minimize some measure of “size” of the residuals

RESIDUALS (unexplained part of variation) are assumed to come from a specific family of random distributions

We have statistical questions (with answers informed by the residuals) about the fitted parameters (like are some of them likely non-zero, or is one different from another?)

Practicing Discovery/Inspection/Use

```
# Although this is sort-of jumping ahead, if you have had some
# statistics before, perhaps your reaction to  df  was to think
# "linear models / regression / ANOVA".  So maybe we know the
# word "ANOVA" for a technique we are interested in, but don't
# know any ways to get R to do it
```

df$obs =
(37-dim. column vector)

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \dots & \dots \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ \dots & \dots \\ 1 & 1 \end{bmatrix} \begin{bmatrix} baselineIntercept \\ treatedEffect \end{bmatrix} + \text{residuals}$$

(37-by-2 "model matrix")

(2-dim. parameter column vector)

We are interested in, e.g., how strong evidence is for *treatedEffect* ≠ 0 ?

(37-dim. column vector; fitting will minimize Euclidean length… ***random model is independent, identically-distributed normal entries***)

…WE ARE ASSUMING A SINGLE "STRATUM" FOR VARIANCE: that it is same across groups… more on this later on

A fit measure like R^2 gives the fraction decrease in the sum-of-squares of the residuals when going from one model (such as constant prediction for all observations) to another (such as the fitted model)

df$grp:

```
"ctrl" -> [ 1 0 ]
"trtd" -> [ 1 1 ]
```

(There are other parameterizations of this simple linear model, but this is the one R will use in this case)

Practicing Discovery/Inspection/Use

```
# Although this is sort-of jumping ahead, if you have had some
# statistics before, perhaps your reaction to  df  was to think
# “linear models / regression / ANOVA”.  So maybe we know the
# word “ANOVA” for a technique we are interested in, but don’t
# know any ways to get R to do it:
??anova          # same as  help.search('anova');  (per ?'??')
# If you just want to search default packages only:
help.search('anova', package=c(getOption('defaultPackages', 'base')))
# Maybe  stats::anova  is what we are looking for?
?stats::anova  # anova(object, …)  and docs. say the  ‘object’
               #   is a “fitted model”, like from “lm” or “glm”
getAnywhere('lm')
A single object matching ‘lm’ was found
It was found in the following places
  package:stats
  namespace:stats
with value ……  # ‘lm’ also from the  stats  package (good sign)
?stats::lm     # This looks possible, and bottom of help page has
               # a little demo of usage;  let’s try on our data…
model <- lm(obs ~ grp, data=df);
# …whatever that is (probably is the model fitting stage!),
# we think it goes into  anova(…)  (stat. Q’s stage?) as 1st arg:
anova(model)
…
```

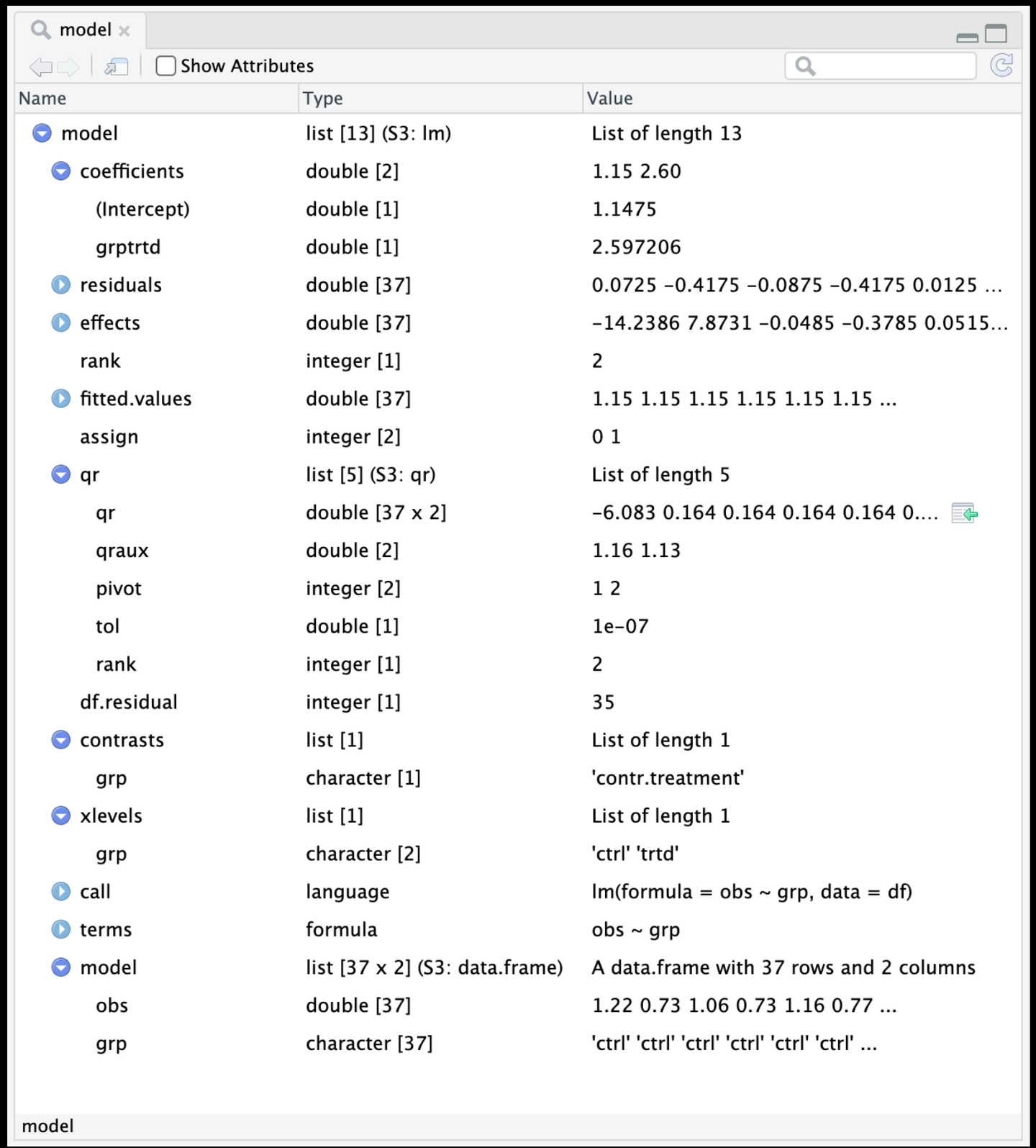
Practicing Discovery/Inspection/Use

```
model <- lm(obs ~ grp, data=df);
# …whatever that is (probably is the model fitting stage!),
# we think it goes into  anova(…)  (stat. Q's stage?) as 1st arg:
anova(model)
Analysis of Variance Table
Response: obs
          Df Sum Sq Mean Sq F value    Pr(>F)
grp        1 61.985  61.985  427.28 < 2.2e-16 ***
Residuals 35  5.077   0.145
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# …that seems to work!  But what is inside  model  and what else
# could we do with it?  Use our R inspection strategies…
```

Inspecting a Fitted Model Object

```
model <- lm(obs ~ grp, data=df);
# …it is an object of class
# 'lm', but is just a length
# 13 list w/ named entries.
# We see stuff in there we
# recognize! – highlights:
#   model$coefficients
#   model$residuals
#   model$fitted.values
# Test understanding:
(r <- model$residuals)
(s <- df$obs -
       model$fitted.values)
s - r
max(abs(s - r))
  # [1] 1.110223e-16
```



Using an Unfamiliar R Object

```
getAnywhere('lm')                                  # lm(…)  function we used comes
A single object matching ‘lm’ was found            # from package  stats  so could try
It was found in the following places               # looking at package documentation
  package:stats
  namespace:stats
with value ……
```

```
class(model)            # model  is an object system object,  so can
[1] "lm"                #  ask R about what method functions are available:
methods(class='lm')
 [1] add1            alias           anova           case.names
 [5] coerce          confint         cooks.distance  deviance
 [9] dfbeta          dfbetas         drop1           dummy.coef
[13] effects         extractAIC      family          formula
[17] hatvalues       influence       initialize      kappa
[21] labels          logLik          model.frame     model.matrix
[25] nobs            plot            predict         print
[29] proj            qr              residuals       rstandard
[33] rstudent        show            simulate        slotsFromS3
[37] summary         variable.names  vcov
see '?methods' for accessing help and source code
# …usually there are more functions you can use an object with
# than this shows  (there are more parts to R’s object systems
# than this, and many functions outside of the object systems),
# but often these are a good start if you have an object output
# and don’t know what you can do with it.

attr(methods(class='lm'), 'info')
« …more detailed, also hints  stats  package probably good to look though… »
```

Using an Unfamiliar R Object

```
anova(model)
Analysis of Variance Table
Response: obs
          Df Sum Sq Mean Sq F value    Pr(>F)
grp        1 61.985  61.985  427.28 < 2.2e-16 ***
Residuals 35  5.077   0.145
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


confint(model)
                2.5 %   97.5 %
(Intercept) 0.9746017 1.320398
grptrtd     2.3421313 2.852280


formula(model)
obs ~ grp
                    # accessor functs. are preferred over directly
residuals(model)    #  taking apart an object when you have a choice
          1           2           3           4           5           6
 0.07250000 -0.41750000 -0.08750000 -0.41750000  0.01250000 -0.37750000
…
         37
 0.13529412
```

Using an Unfamiliar R Object

```
cooks.distance(model)
            1             2             3             4             5
1.003681e-03 3.328377e-02 1.461961e-03 3.328377e-02 2.983593e-05
…
           36            37
1.169124e-01 4.189513e-03


vcov(model)
             (Intercept)      grptrtd
(Intercept)  0.007253426 -0.007253426
grptrtd     -0.007253426  0.015786869
```

```
model.matrix(model)
   (Intercept) grptrtd
1            1        0
2            1        0
……           …        …
20           1        0
21           1        1
22           1        1
……           …        …
37           1        1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$grp
[1] "contr.treatment"
```

```
# test our understanding,  %*%  is matrix multiply:
#   (f <- model$fitted.values)
#   (g <- model.matrix(model) %*% model$coefficients)
#   max(abs(f - g))   # [1] 6.661338e-16
```

Using an Unfamiliar R Object

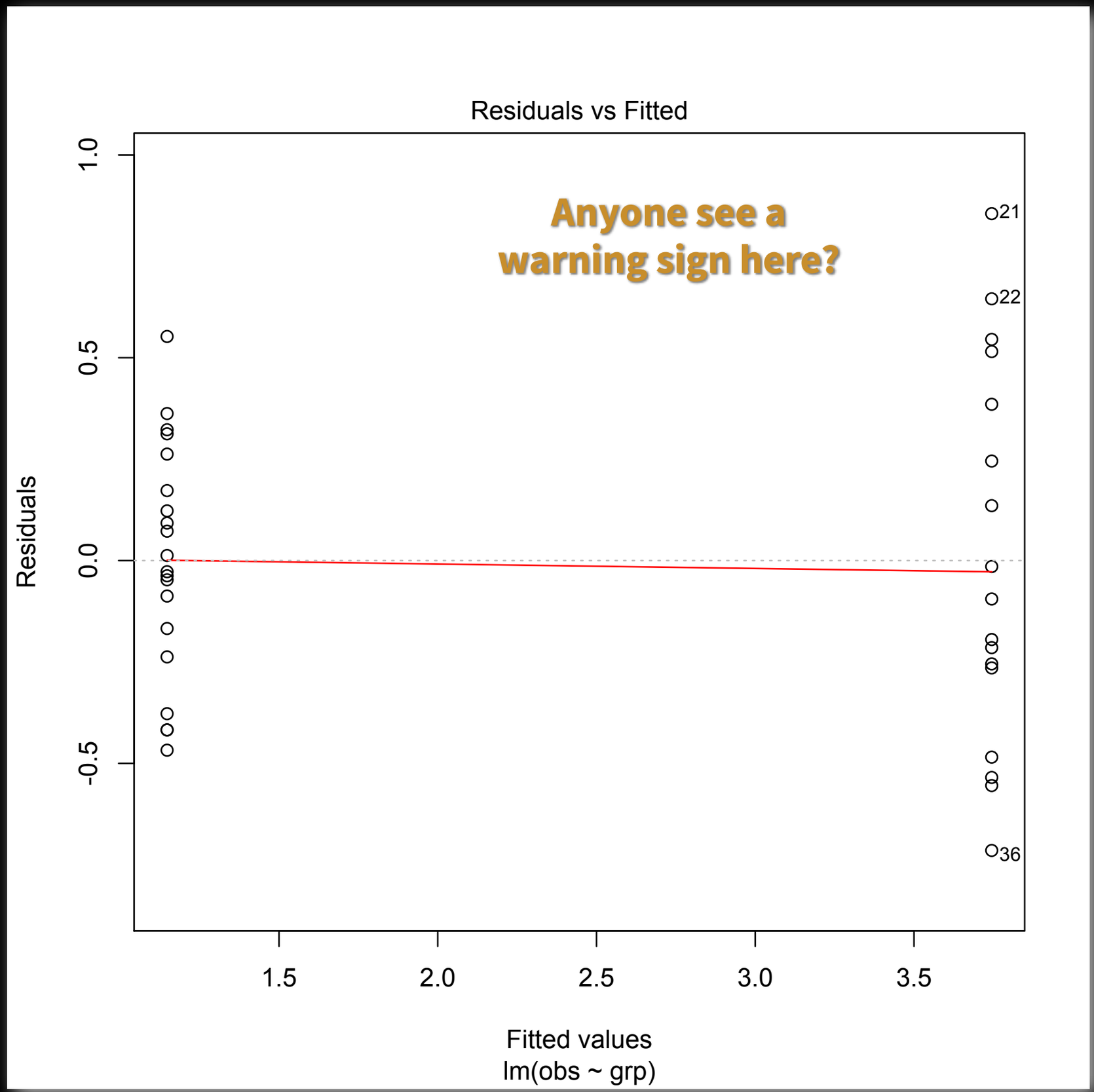
```
summary(model)
Call:
lm(formula = obs ~ grp, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7147 -0.2547 -0.0275  0.2625  0.8553

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.14750    0.08517   13.47 2.05e-15 ***
grptrtd      2.59721    0.12565   20.67  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3809 on 35 degrees of freedom
Multiple R-squared:  0.9243,    Adjusted R-squared:  0.9221
F-statistic: 427.3 on 1 and 35 DF,  p-value: < 2.2e-16
```

```
methods(summary)
 [1] summary.aov                summary.aovlist*
 [3] summary.aspell*            summary.check_packages_in_dir*
 [5] summary.connection         summary.data.frame
 [7] summary.Date               summary.default
 [9] summary.ecdf*              summary.factor
[11] summary.glm                summary.infl*
[13] summary.lm                 summary.loess*
[15] summary.manova             summary.matrix
[17] summary.mlm*               summary.nls*
[19] summary.packageStatus*     summary.POSIXct
[21] summary.POSIXlt            summary.ppr*
[23] summary.prcomp*            summary.princomp*
[25] summary.proc_time          summary.srcfile
[27] summary.srcref             summary.stepfun
[29] summary.stl*               summary.table
[31] summary.tukeysmooth*       summary.warnings

see '?methods' for accessing help and source code
```

```
# …try  methods(print)  to see how many specializations that has!
```

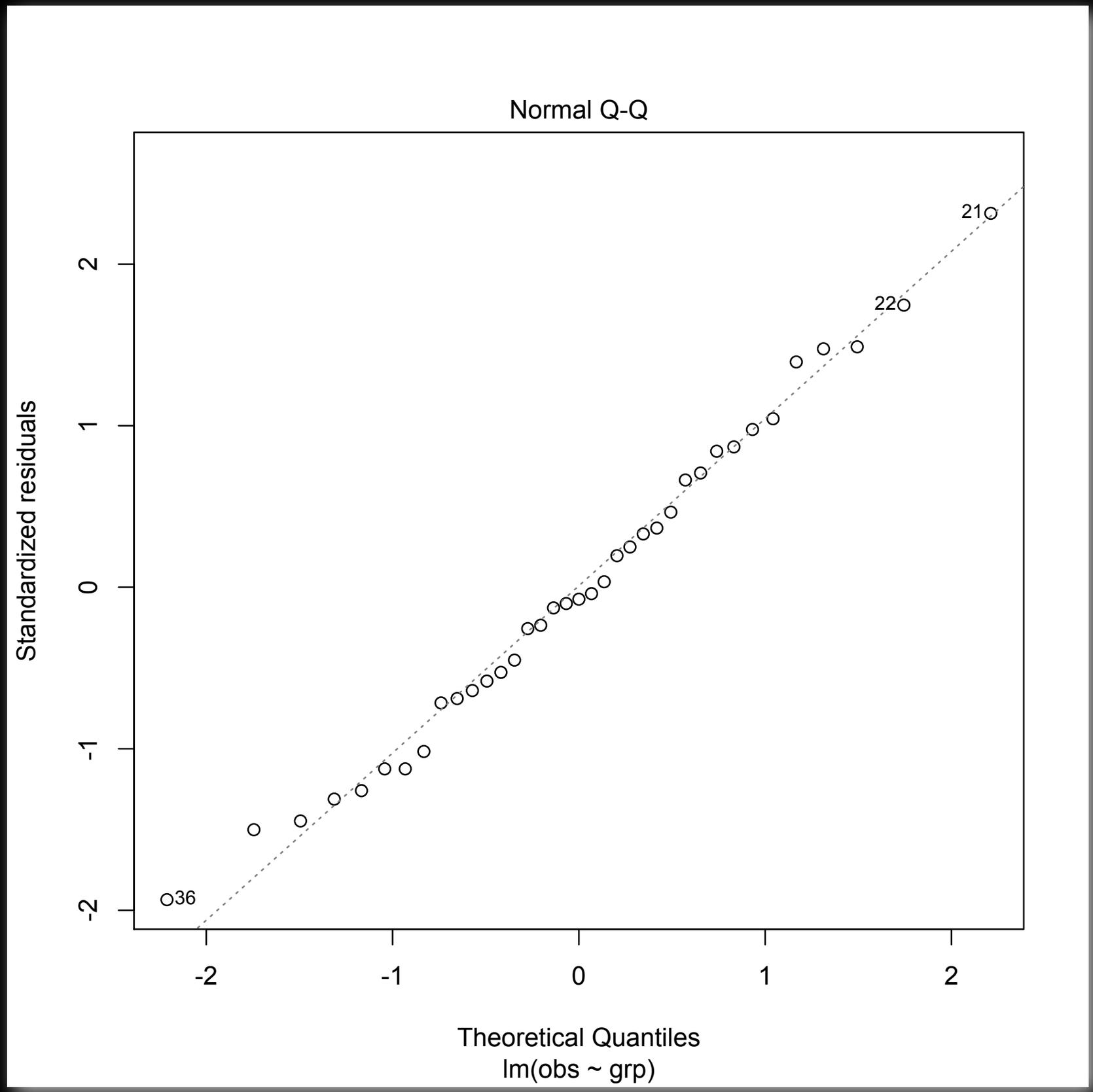
Using an Unfamiliar R Object

```
plot(model)
Hit <Return> to see
next plot:
```



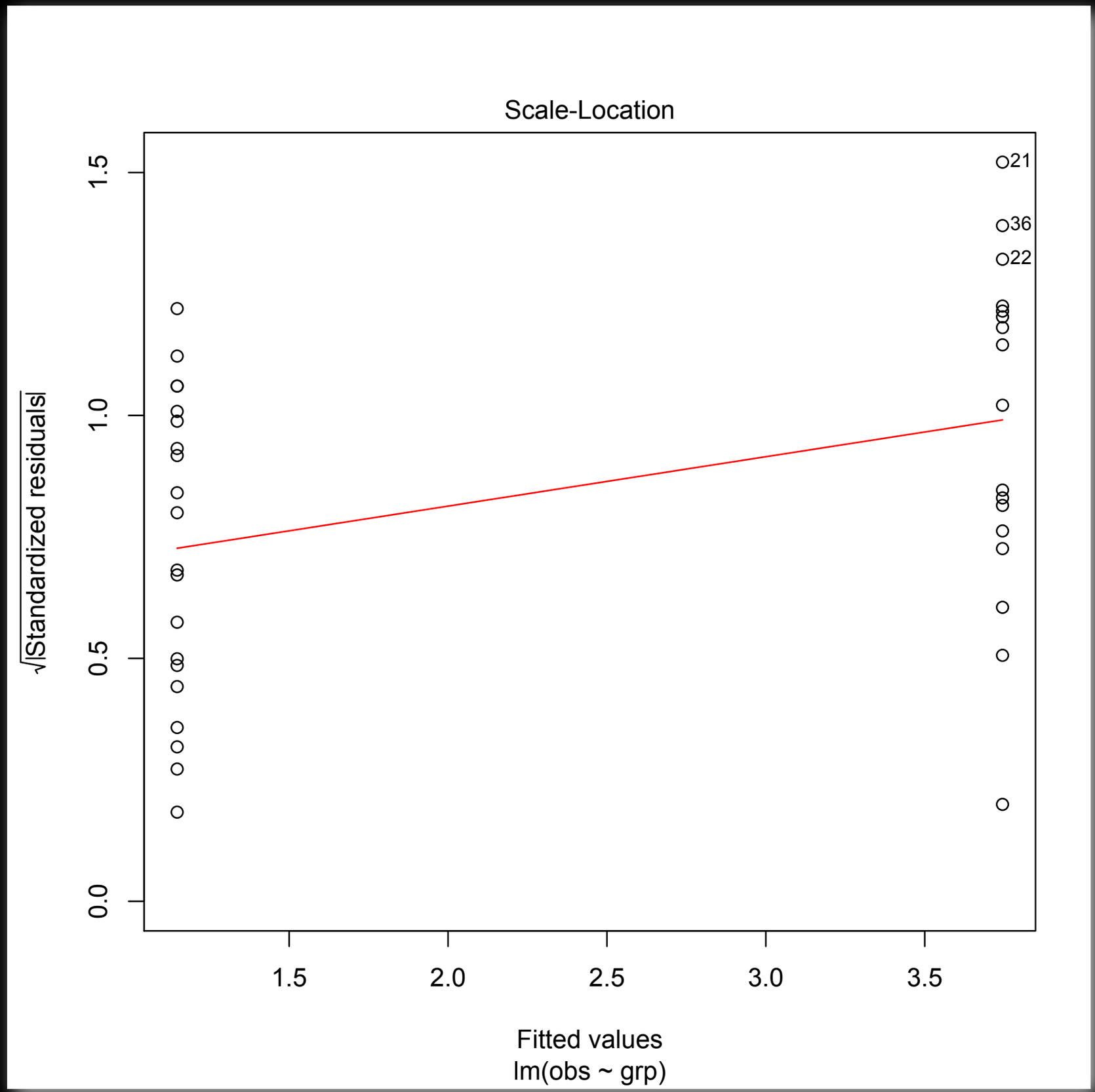
Using an Unfamiliar R Object

```
plot(model)
Hit <Return> to see
next plot:
Hit <Return> to see
next plot:
```



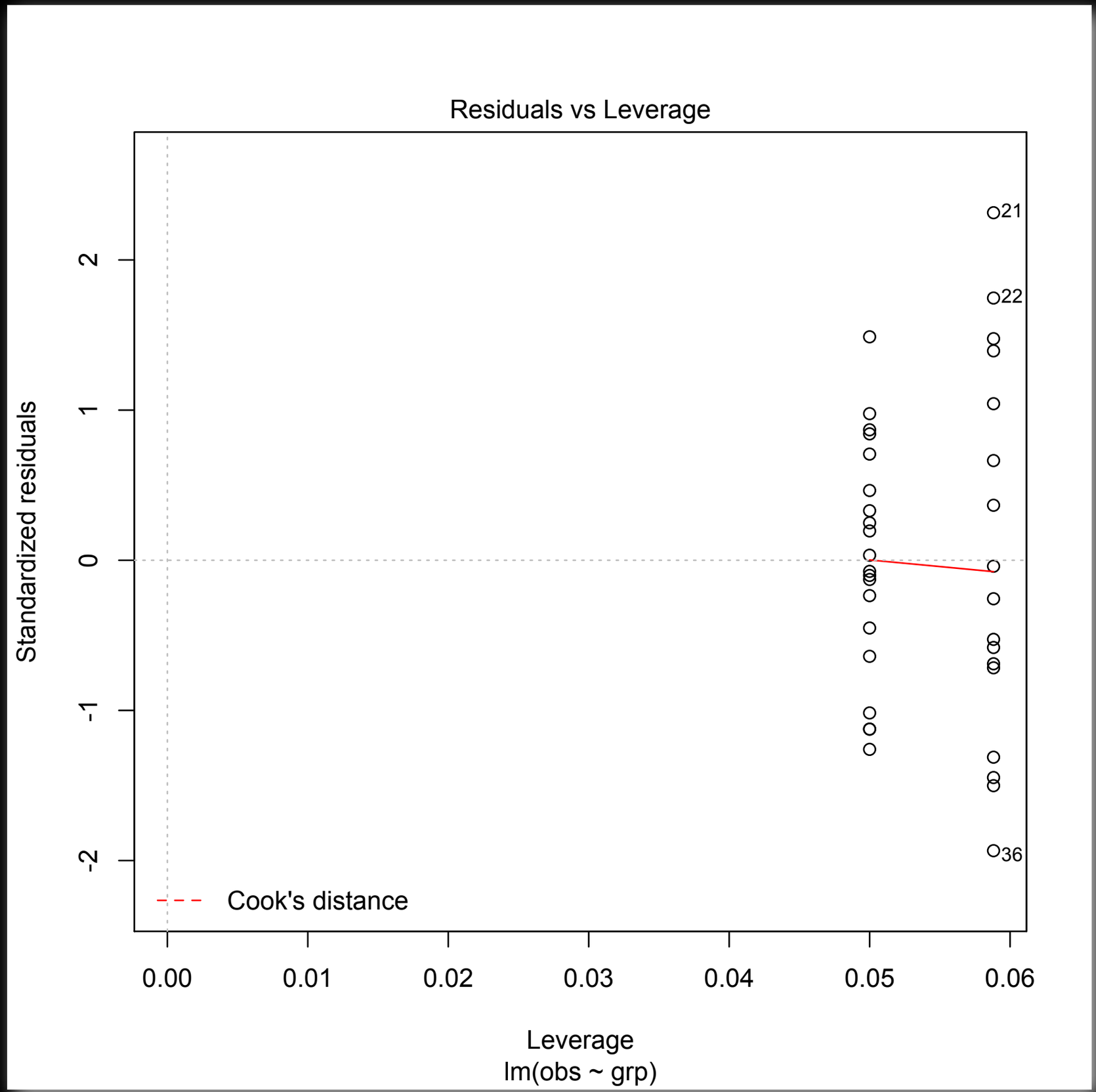
Using an Unfamiliar R Object

```
plot(model)
Hit <Return> to see
next plot:
Hit <Return> to see
next plot:
Hit <Return> to see
next plot:
```

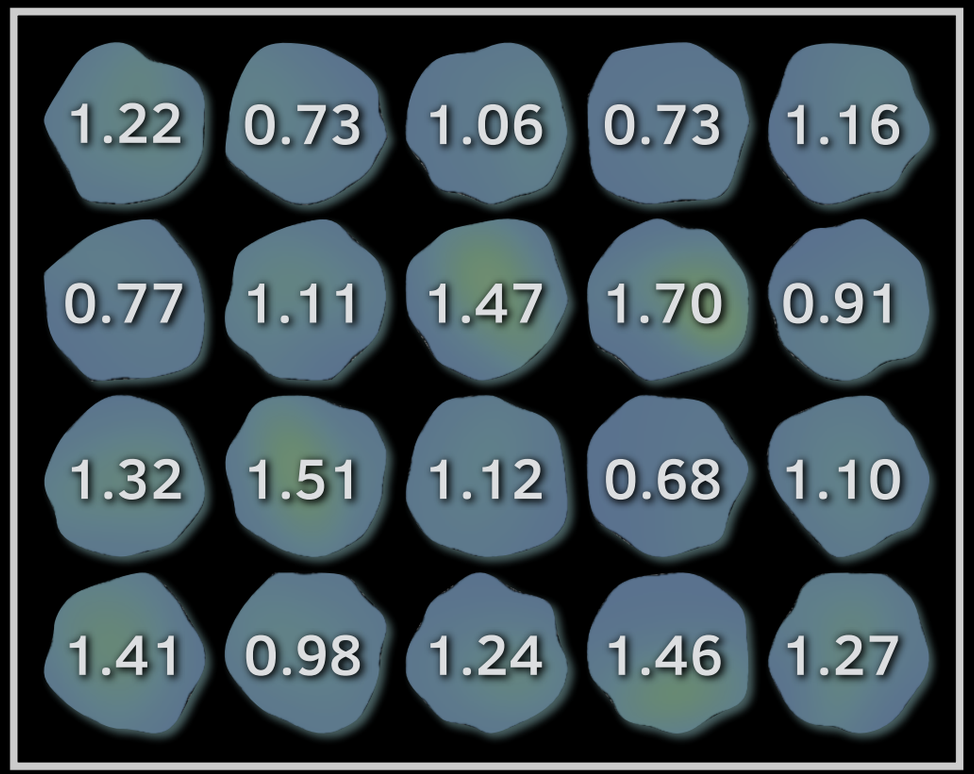


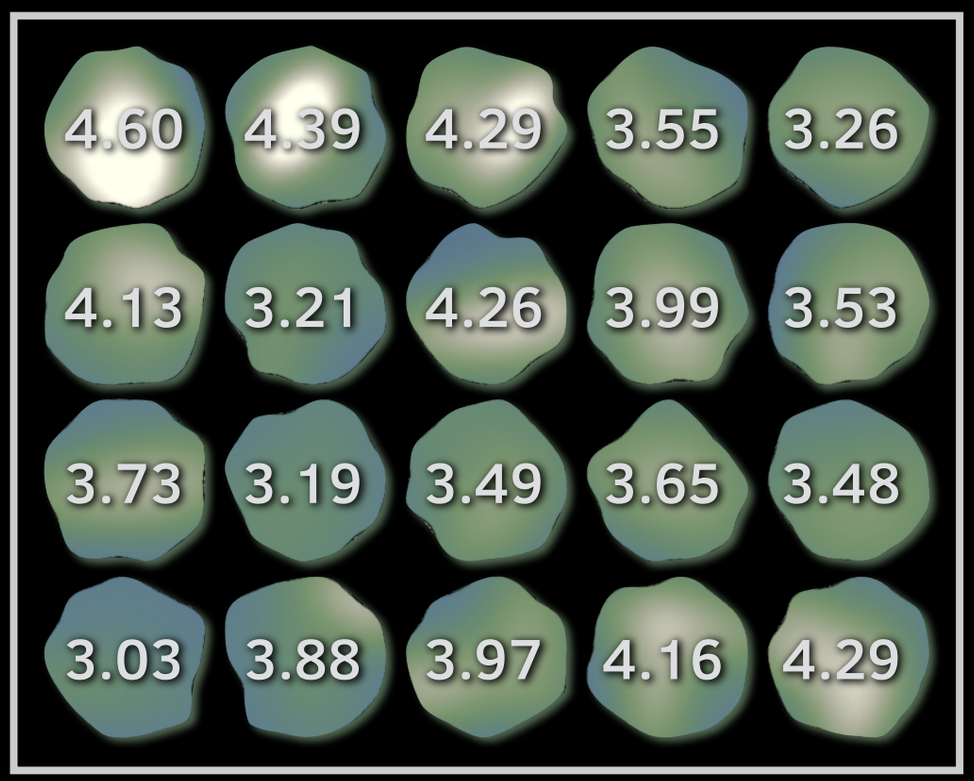
Using an Unfamiliar R Object

```
plot(model)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:

stats:::plot.lm
« …implementing R code, e.g., if you wanted to see how one of the plots was made, maybe so you could modify it… »
```



Back to Basics, for real this time!

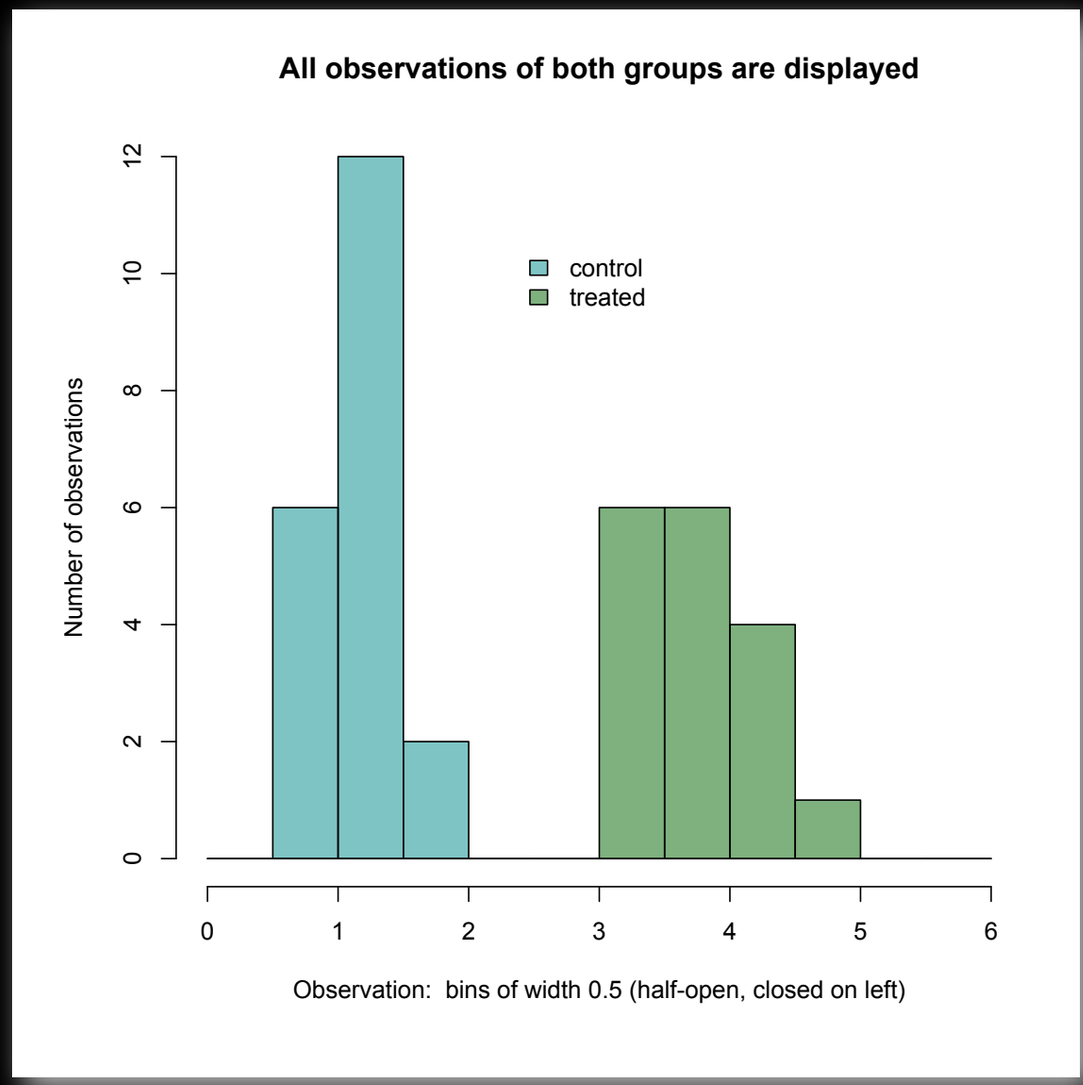





```
R> (df <- data.frame(obs=c(ctrl, trtd),
                     grp=c(rep("ctrl", length(ctrl)),
                           rep("trtd", length(trtd)) )))
```

```
    obs  grp
1  1.22 ctrl
2  0.73 ctrl
..... ..... .....
19 1.46 ctrl
20 1.27 ctrl
21 4.60 trtd
22 4.39 trtd
..... ..... .....
36 3.03 trtd
37 3.88 trtd
```

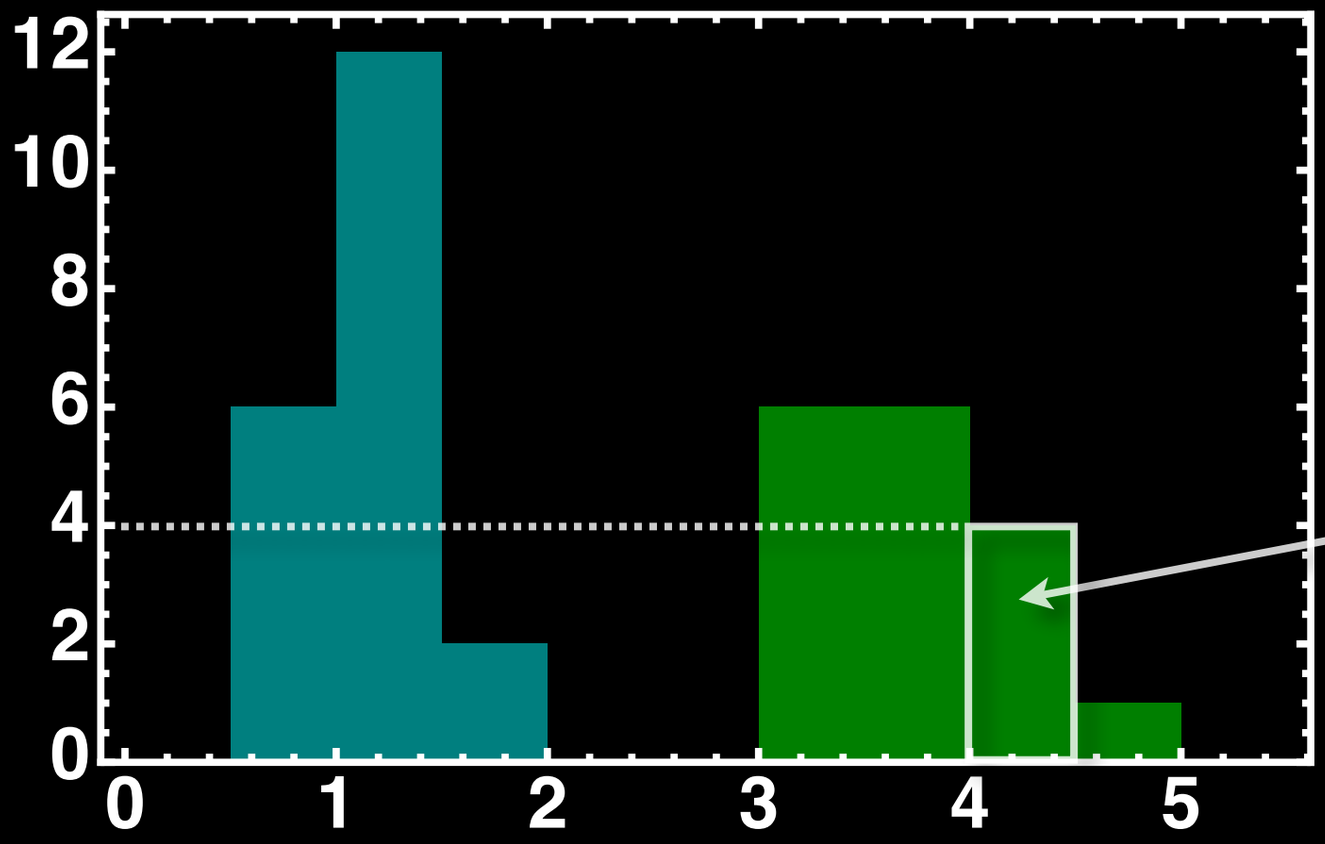
BTW, data.frame's use *row.names* and *names* attributes for row and column names, resp.



Philosophy

- There are various philosophical interpretations of probability and statistics… (not going to get into)
  - ***In Experimental Biology, we are mostly concerned with experiments that could be "REPLICATED"***
    - Outcome of an experiment — even "decision" to conduct an experiment — could depend on the entire state of the universe (and we are part of it)
      - *…however all but the merest sliver of that state presumably completely irrelevant ("**independent**"); we could "**REPEAT**" an experiment over and over*
        - Each repeat more or less "fresh" — independent of others
      - We are mostly interested in observational patterns that are (or would be) "stable" under repetition
- **Probability**[something specific happens or not in each repeat] **:= fraction of times it would happen** if repeated experiment forever

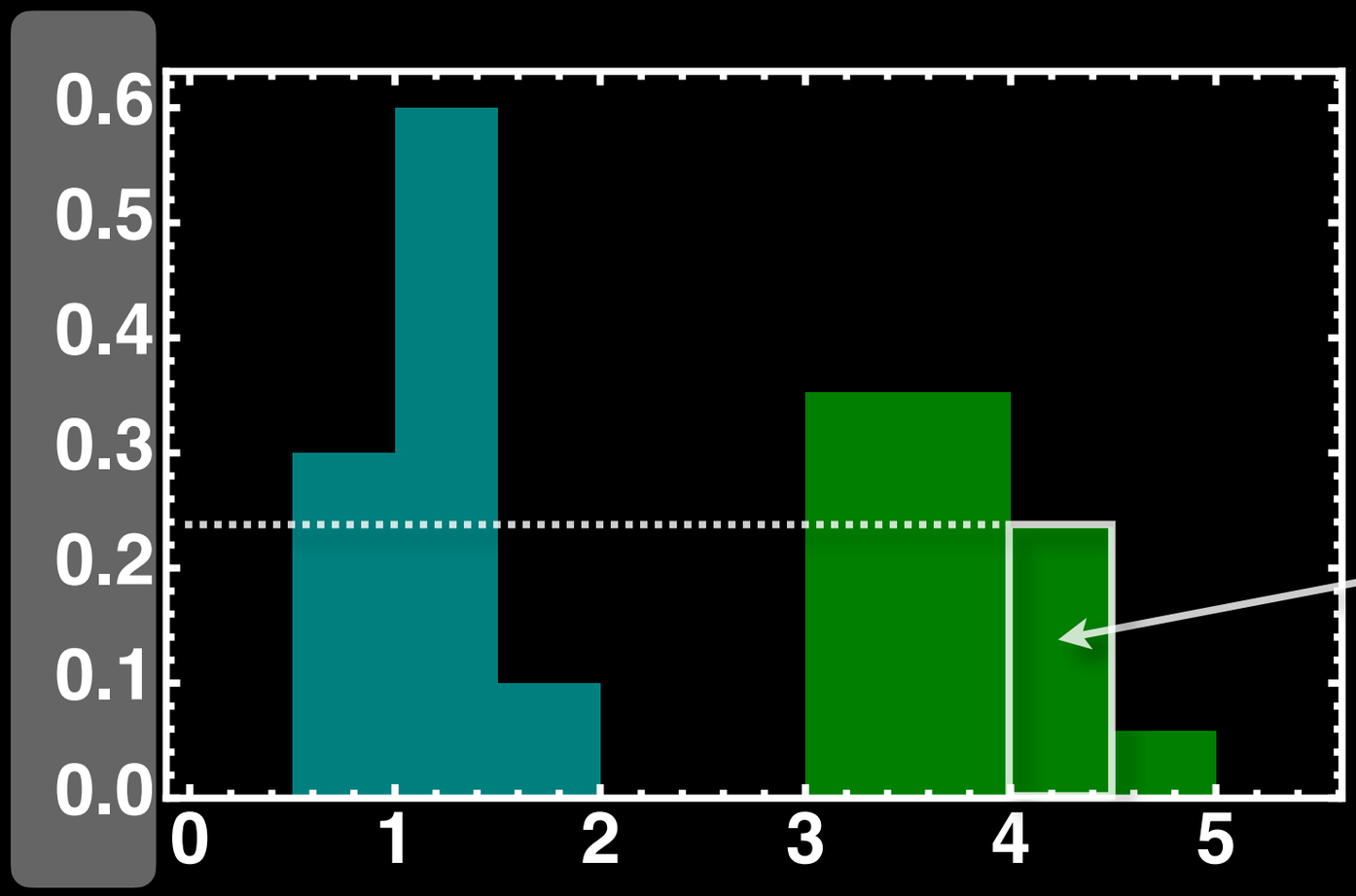
Toward Probability Distributions



Height of this bar shows
NUMBER of observations
landing in interval [4, 4.5)

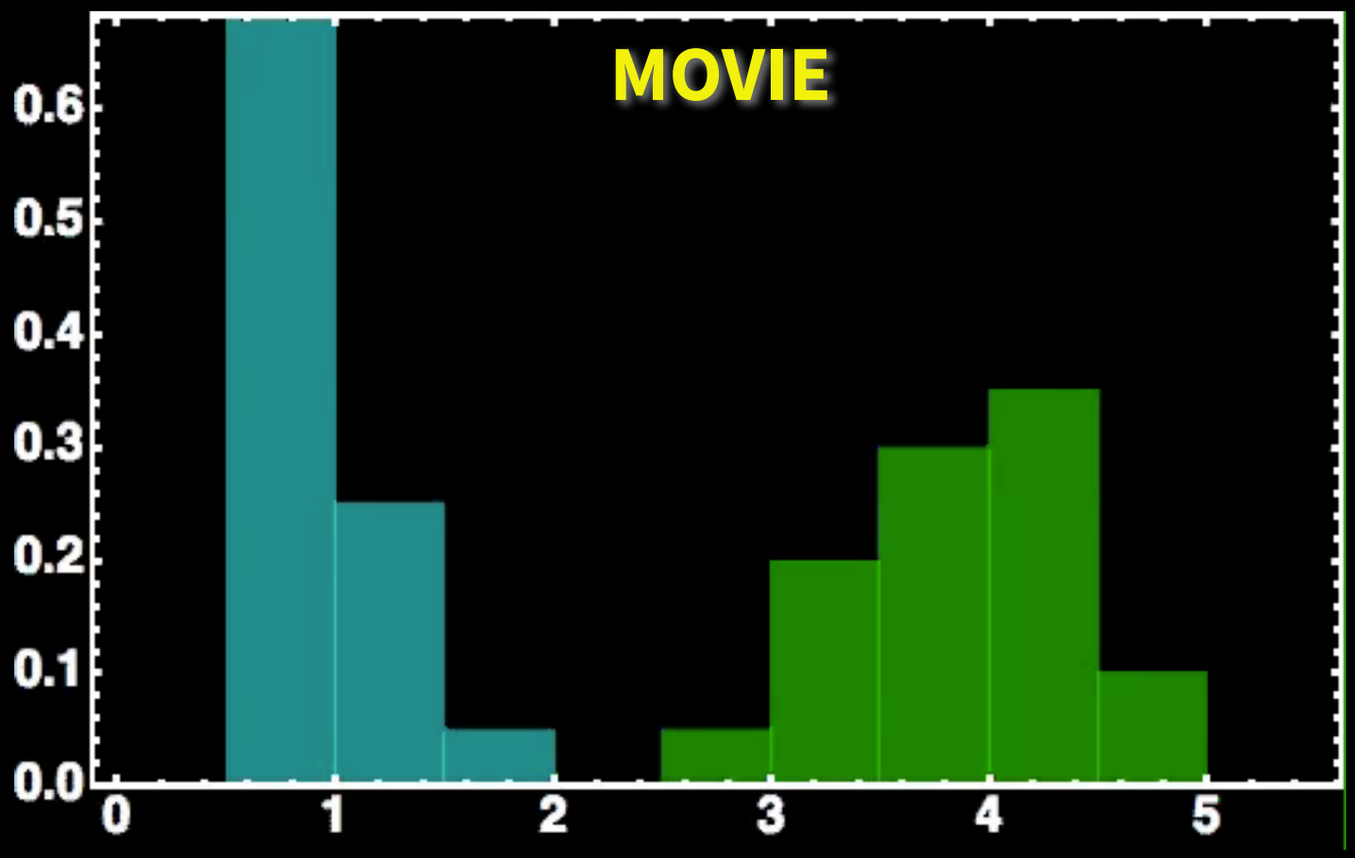
For same reasons mean
is preferred over raw total,
switch y axis to per–group
FRACTION of observations…

Toward Probability Distributions



Height of this bar shows **FRACTION** of observations (within group) **that landed in interval [4, 4.5)**

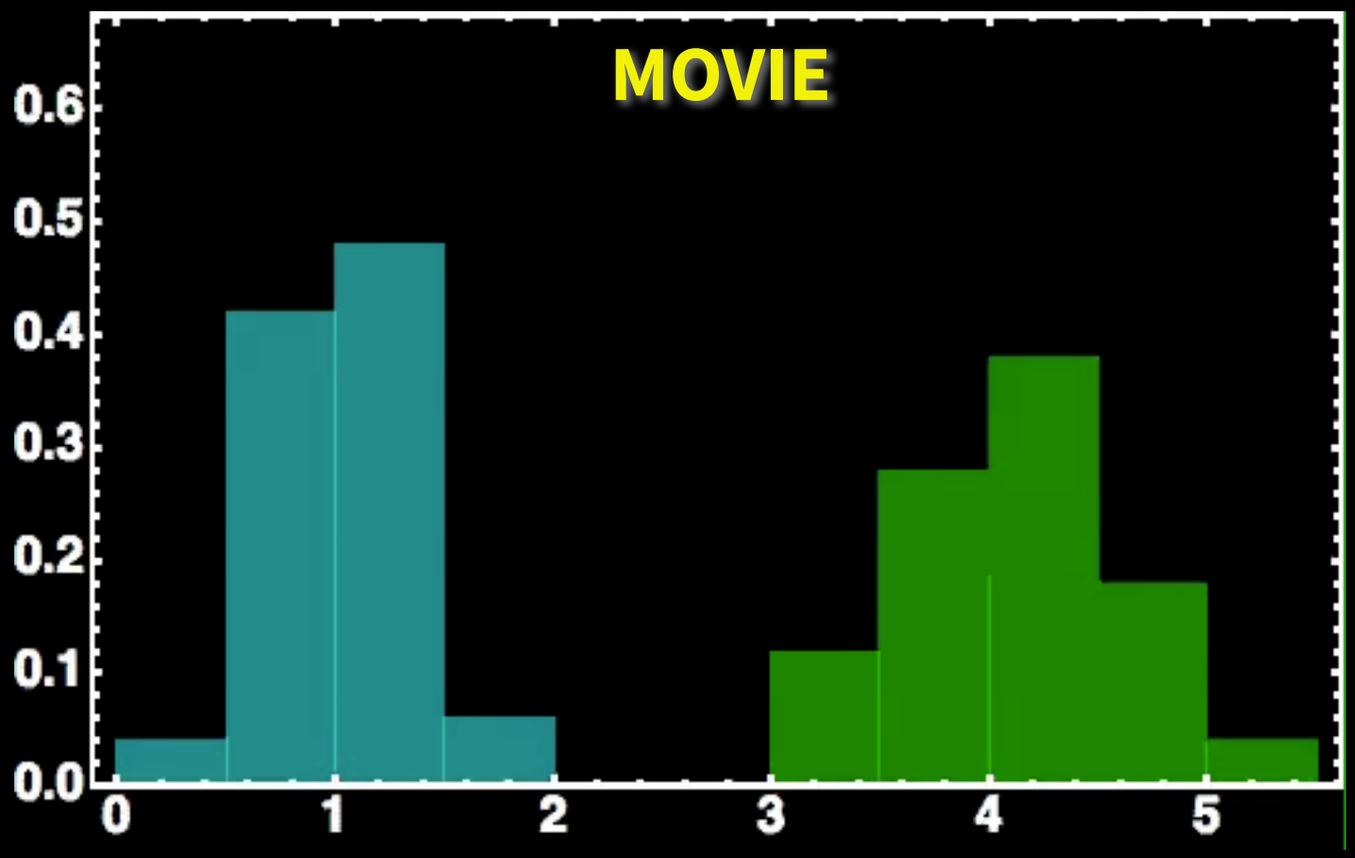
Repeating Whole Experiments



20 observations per group per experiment

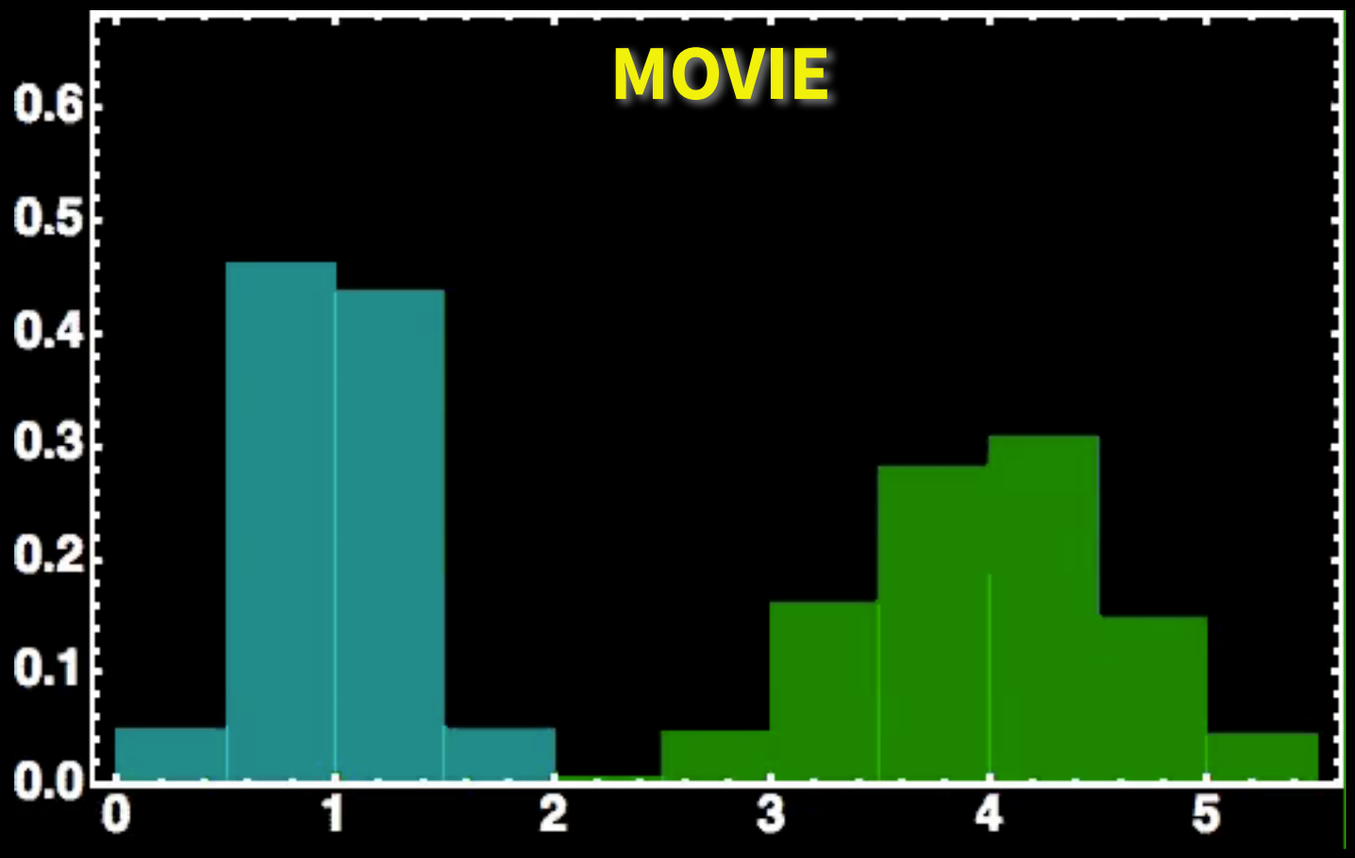
x and *y* axes are kept locked, even though some experiments have observations falling outside *x* limits and/or bars going above the top of *y*; don't worry much about this

Repeating Whole Experiments



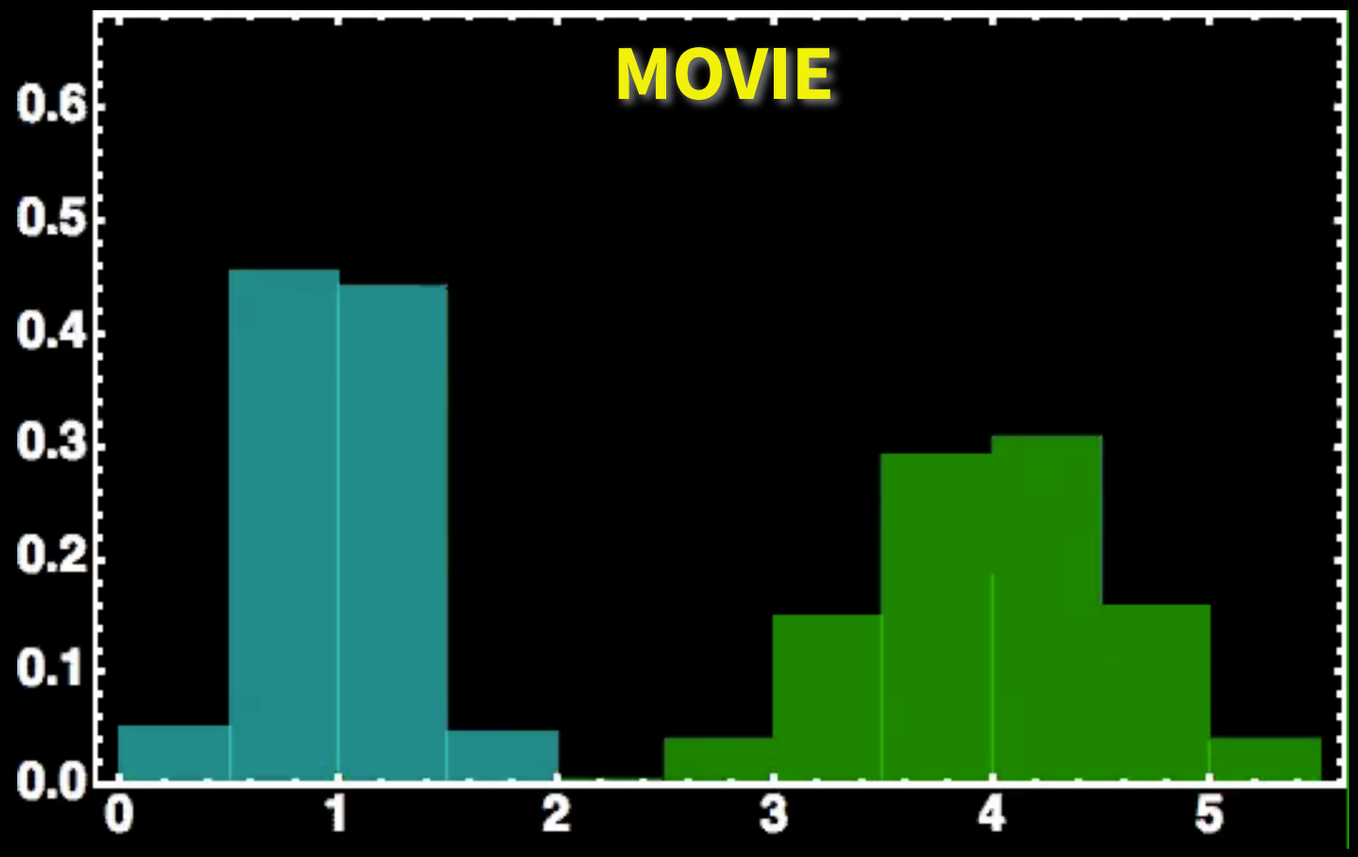
50 observations
per group
per experiment

Repeating Whole Experiments



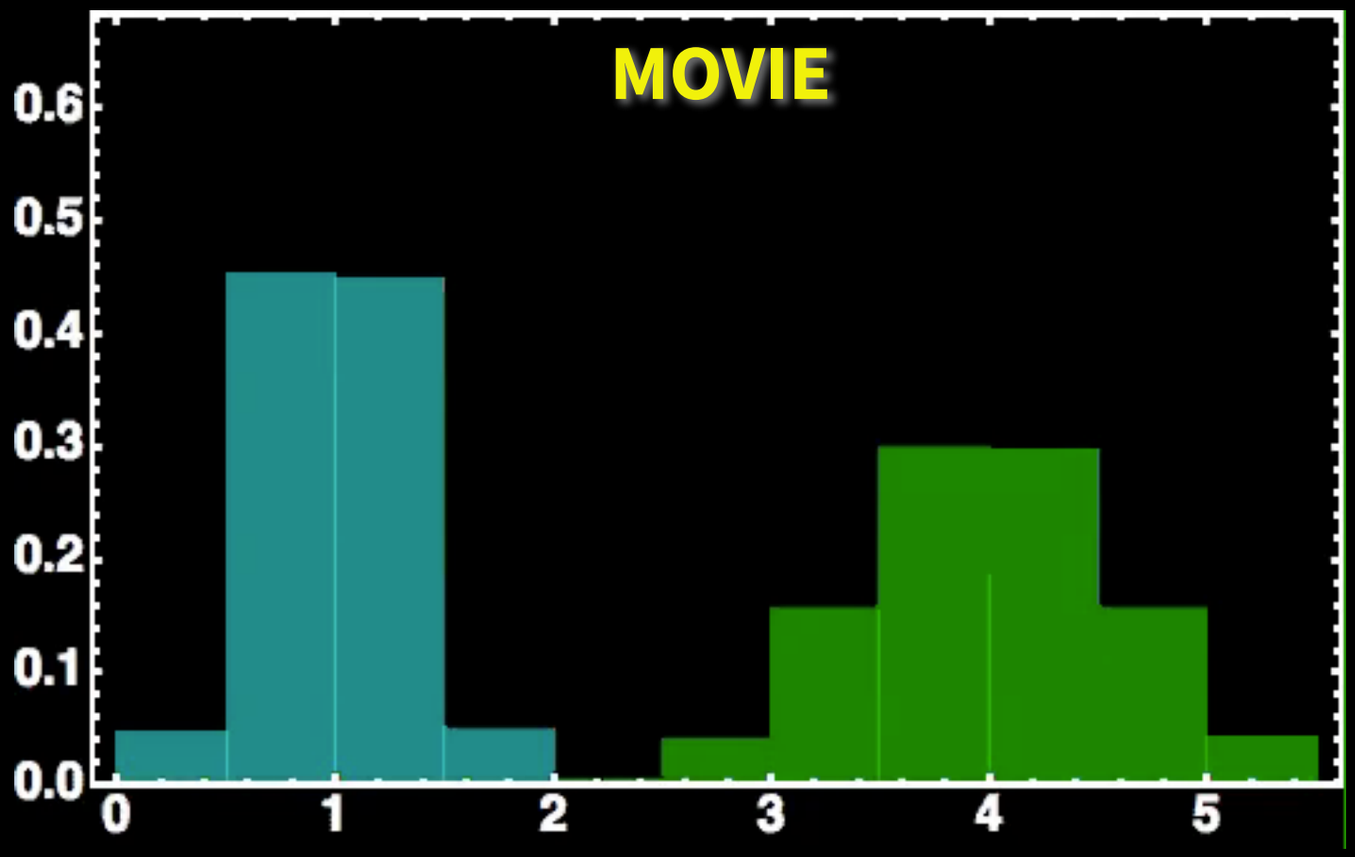
1,000 observations
per group
per experiment

Repeating Whole Experiments



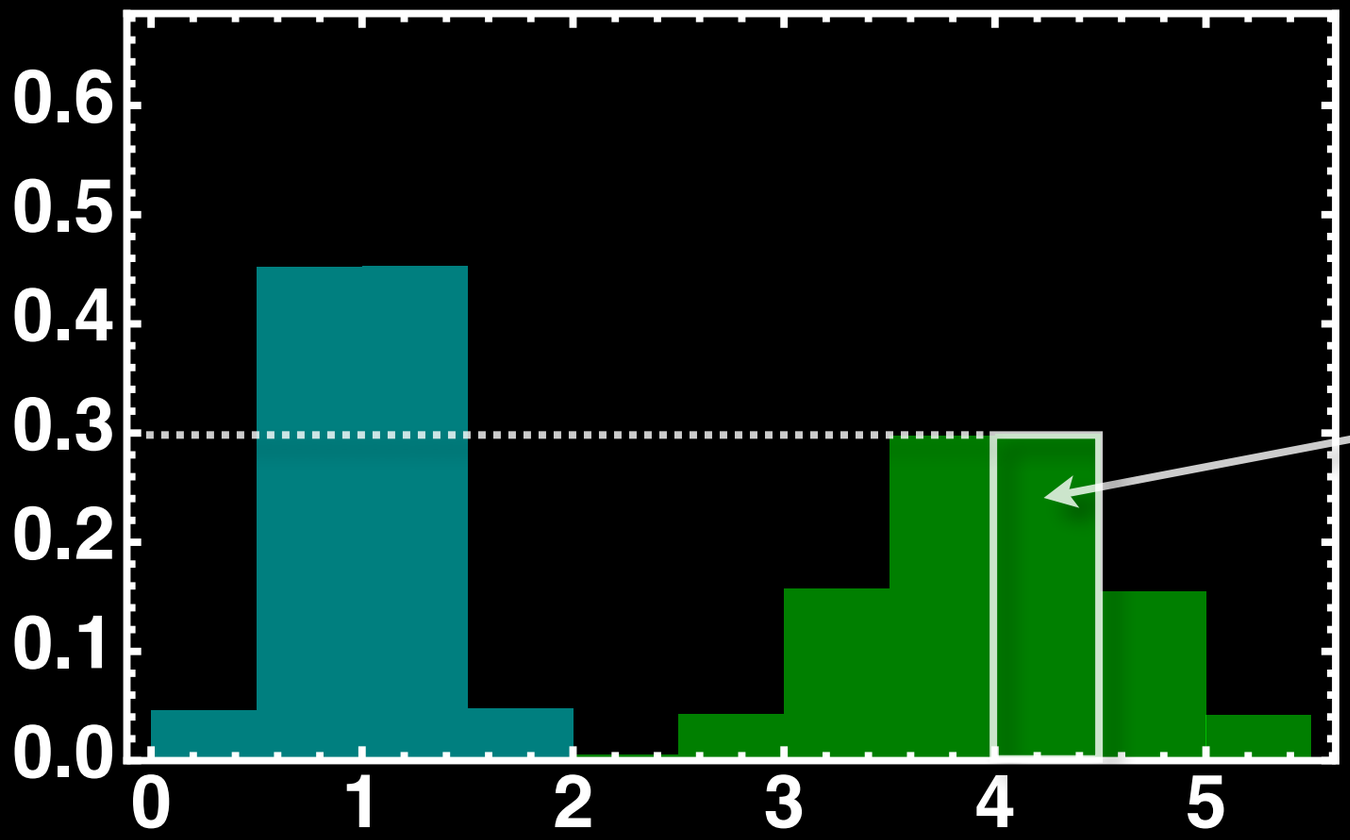
10,000 observations
per group
per experiment

Repeating Whole Experiments



100,000 observations
per group
per experiment

Toward Probability Distributions

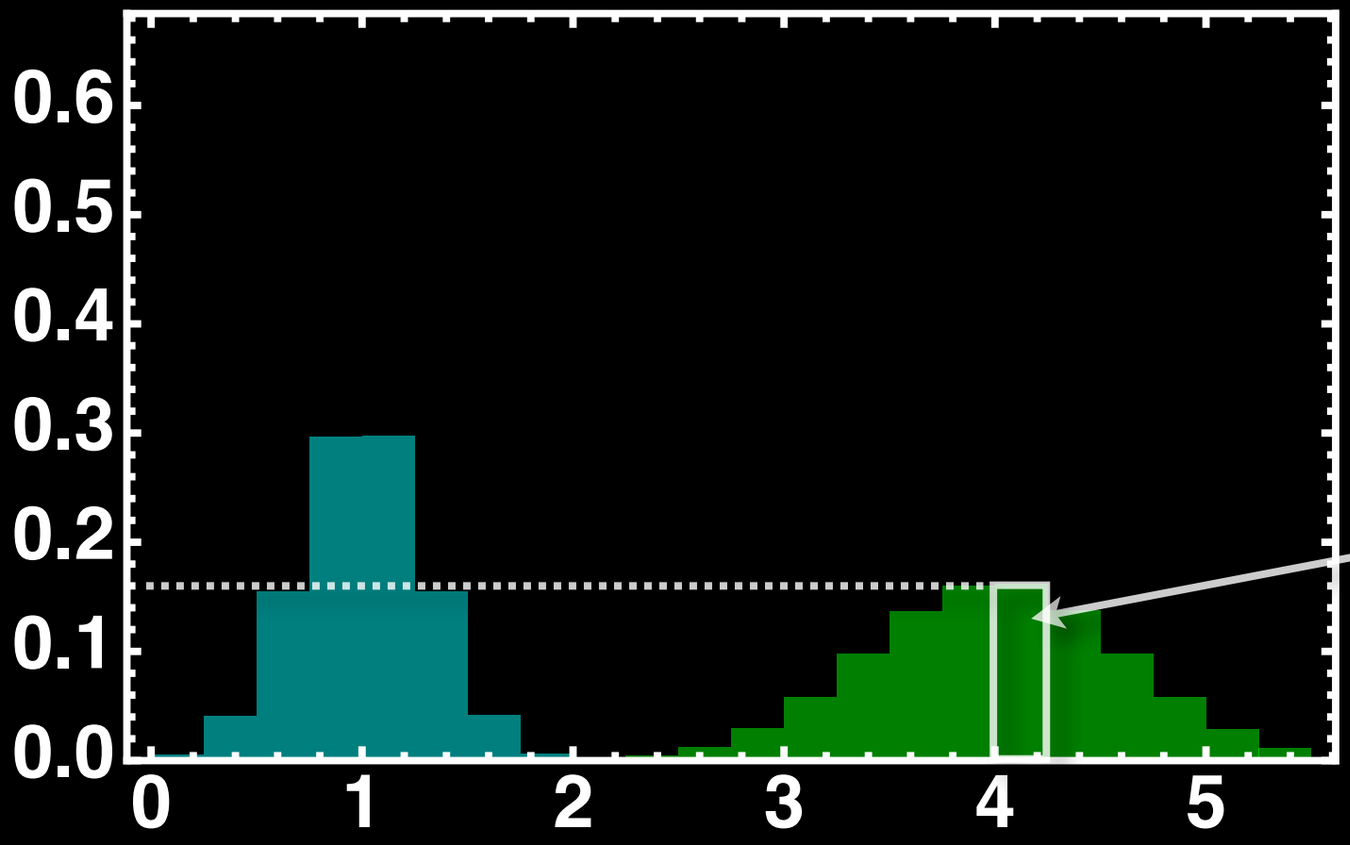


100,000 observations per group per experiment

This fraction is now very close to the **probability** that a treated group observation is in [4, 4.5)

What if interested in higher x axis resolution? — make the bins smaller…

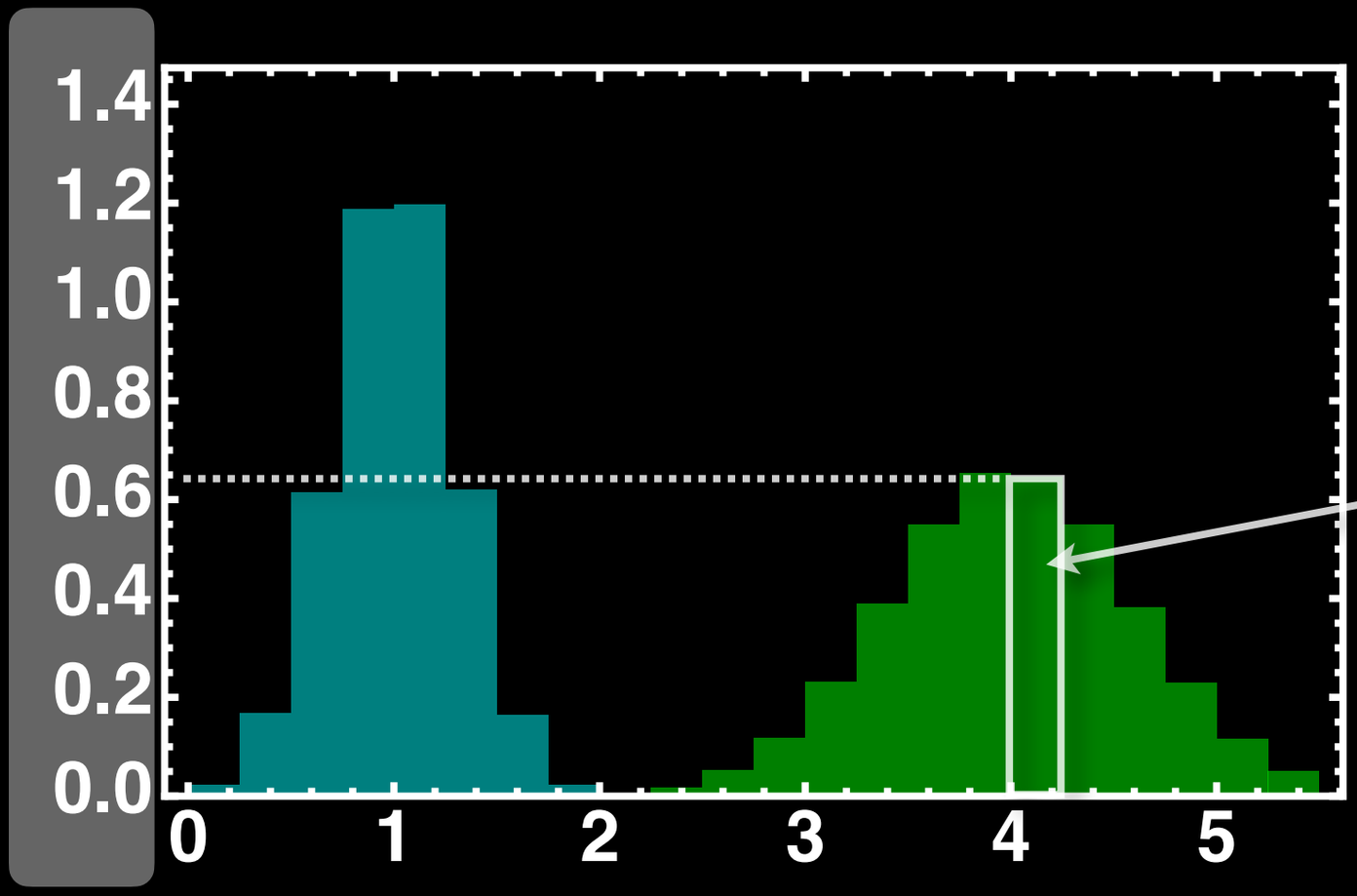
Toward Probability Distributions



100,000 observations per group per experiment

…but bars got more or less half as high…

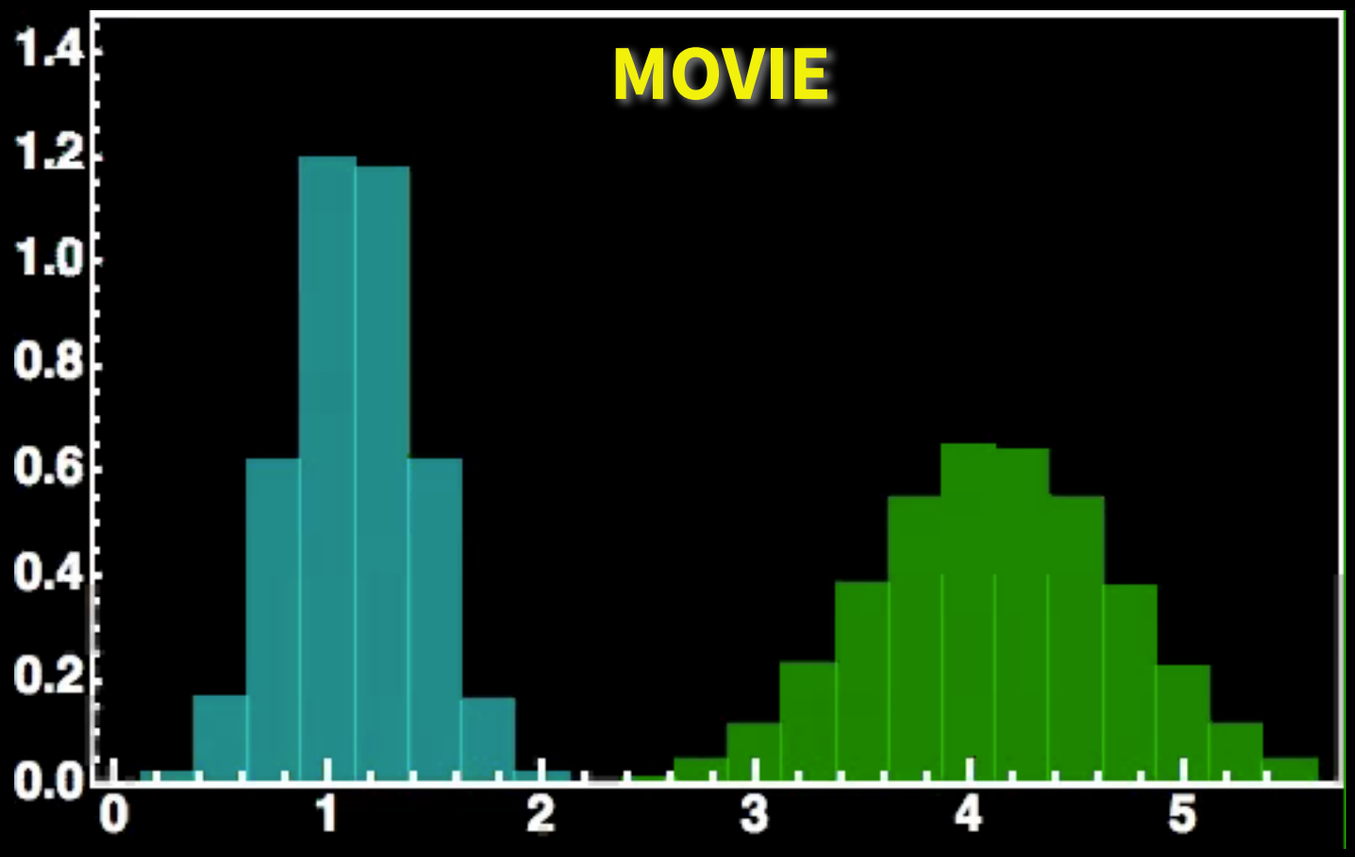
Toward Probability Distributions



100,000 observations per group per experiment

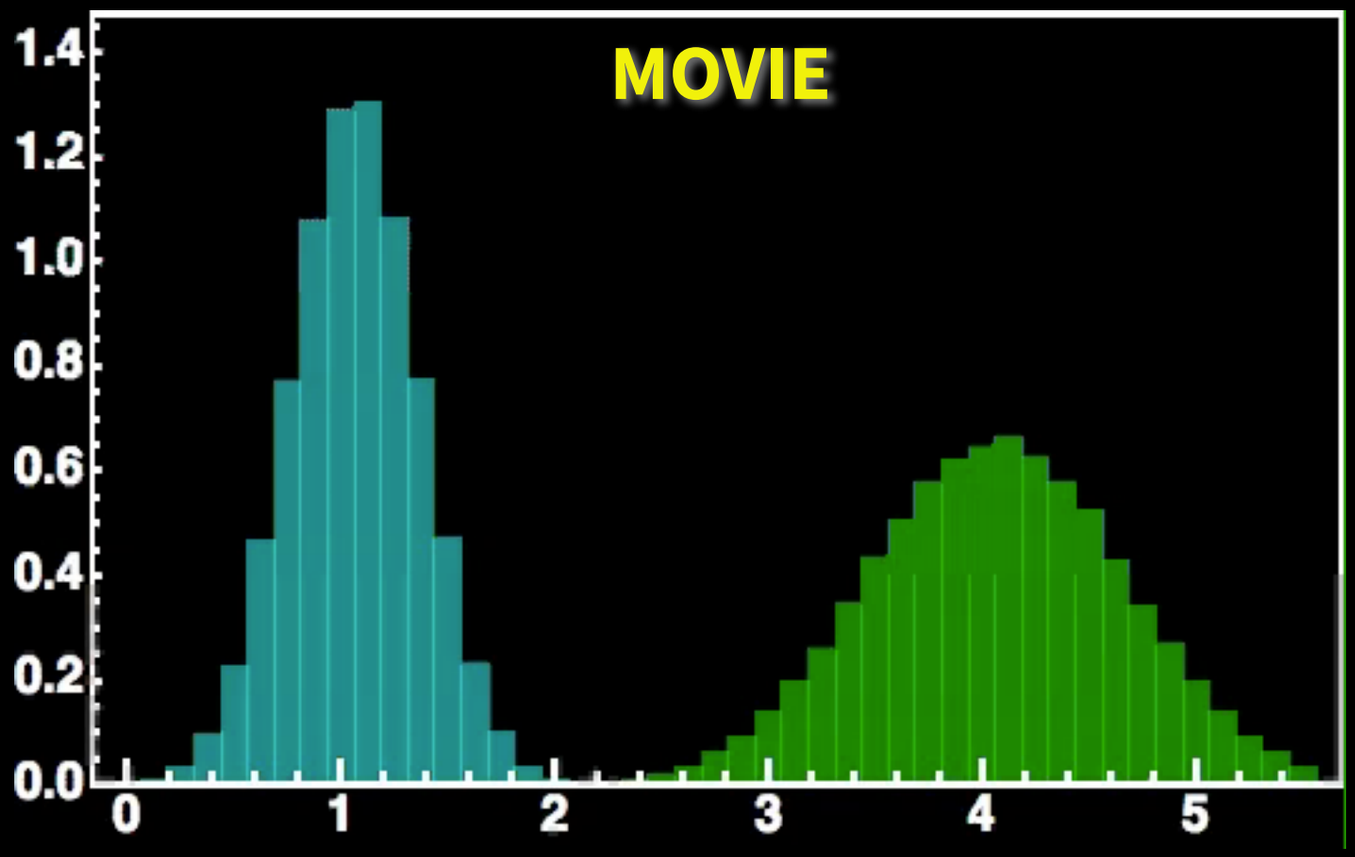
Height of this bar shows DENSITY of observations (within group) **= fraction of observations divided by width of bin** (similar idea as passage from total to mean)

Toward Probability Distributions



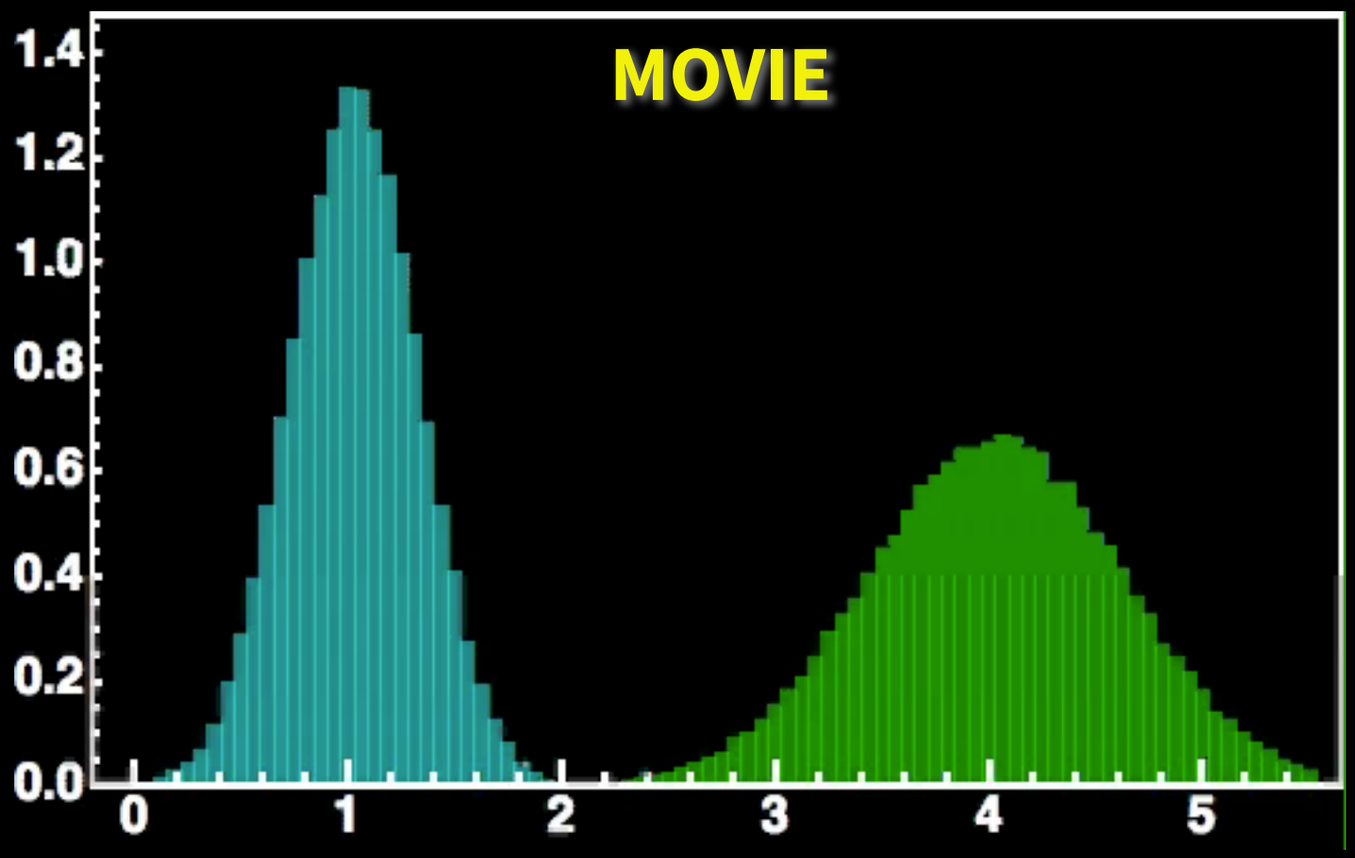
100,000 observations
per group
per experiment,
bins of width 1/4

Toward Probability Distributions



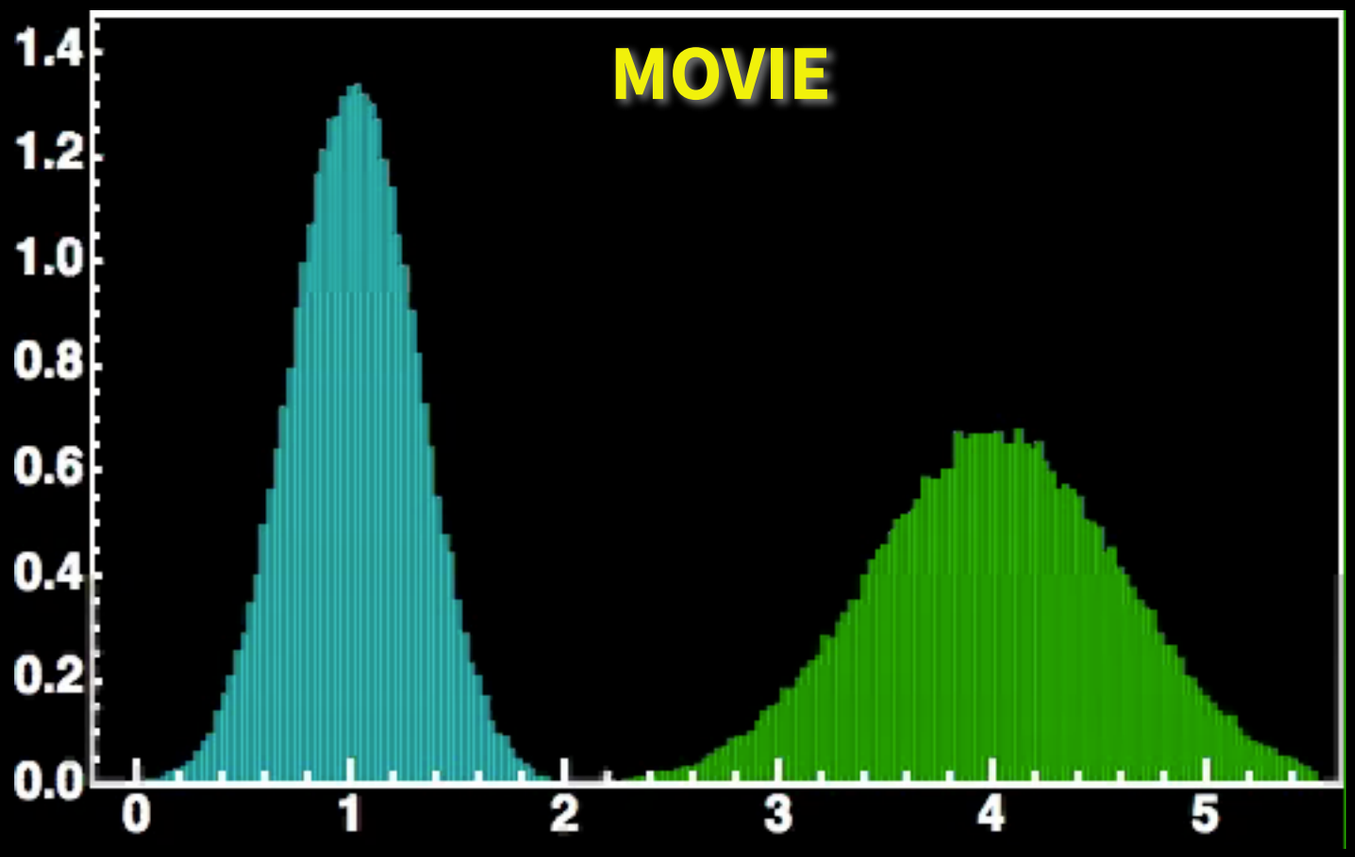
100,000 observations
per group
per experiment,
bins of width 1/8

Toward Probability Distributions



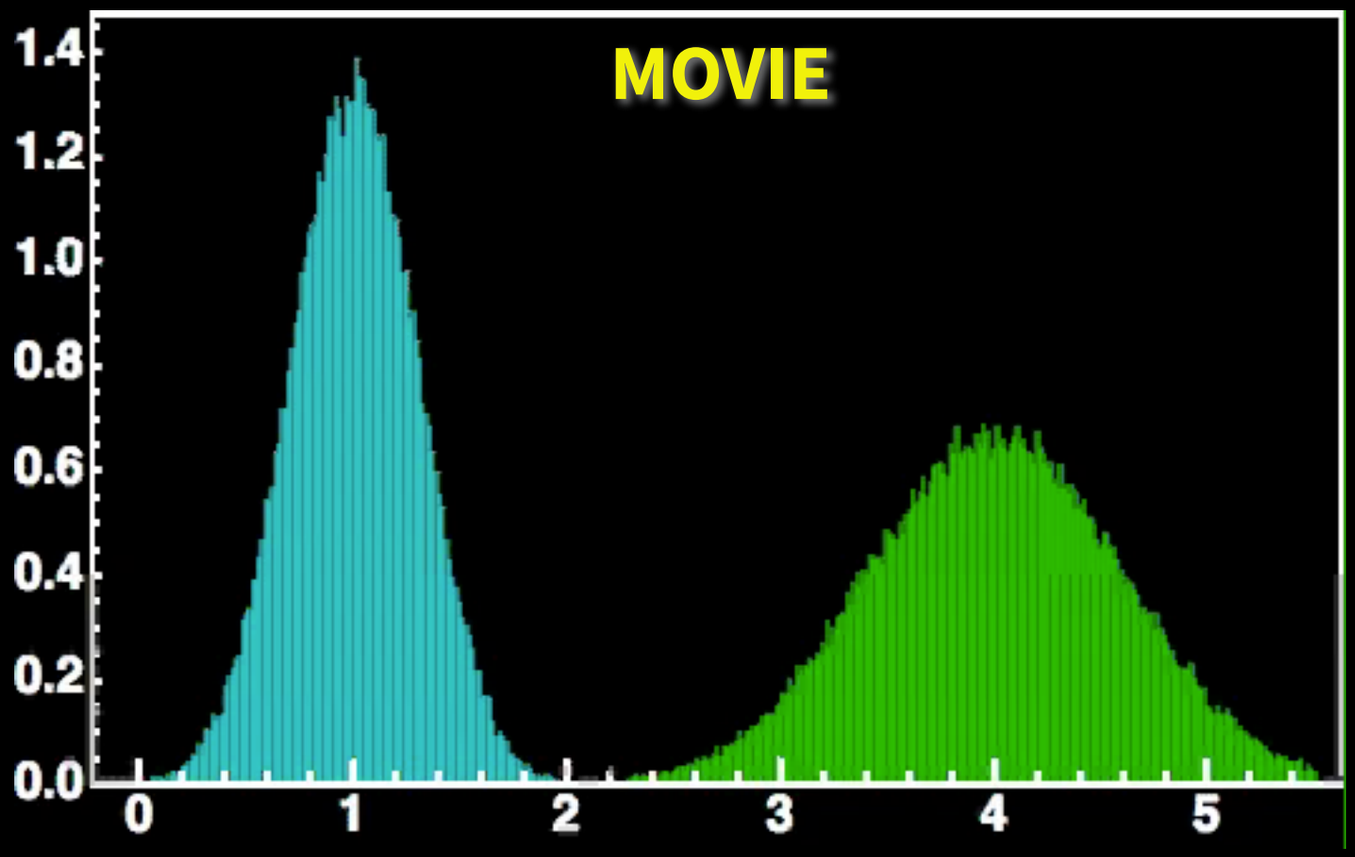
100,000 observations
per group
per experiment,
bins of width 1/16

Toward Probability Distributions



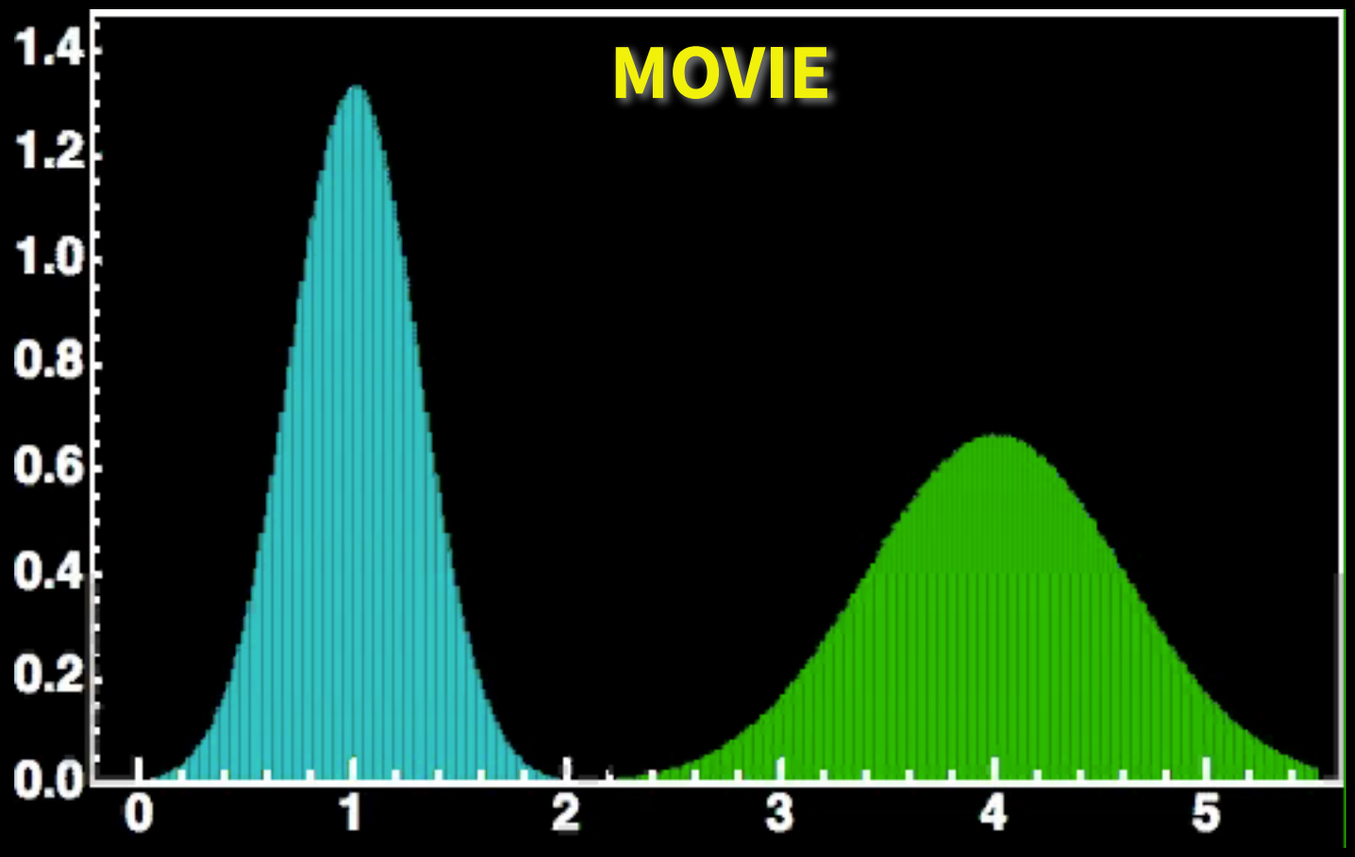
100,000 observations
per group
per experiment,
bins of width 1/32

Toward Probability Distributions



100,000 observations
per group
per experiment,
bins of width 1/64

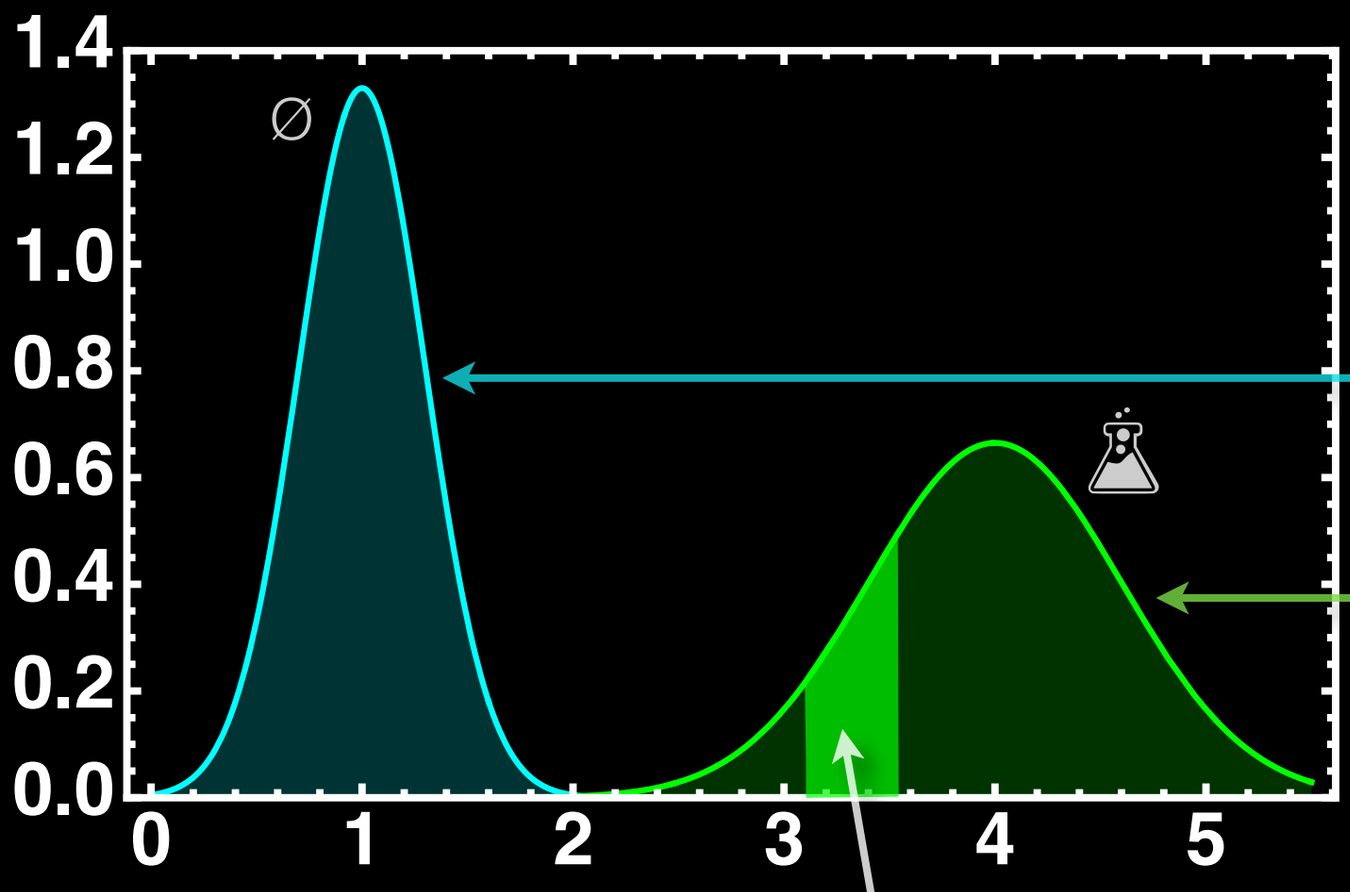
Toward Probability Distributions



10,000,000 observations
per group
per experiment,
bins of width 1/64

Probability Distributions

(continuous “case”)



Infinity observations per group per experiment, bins of width zero

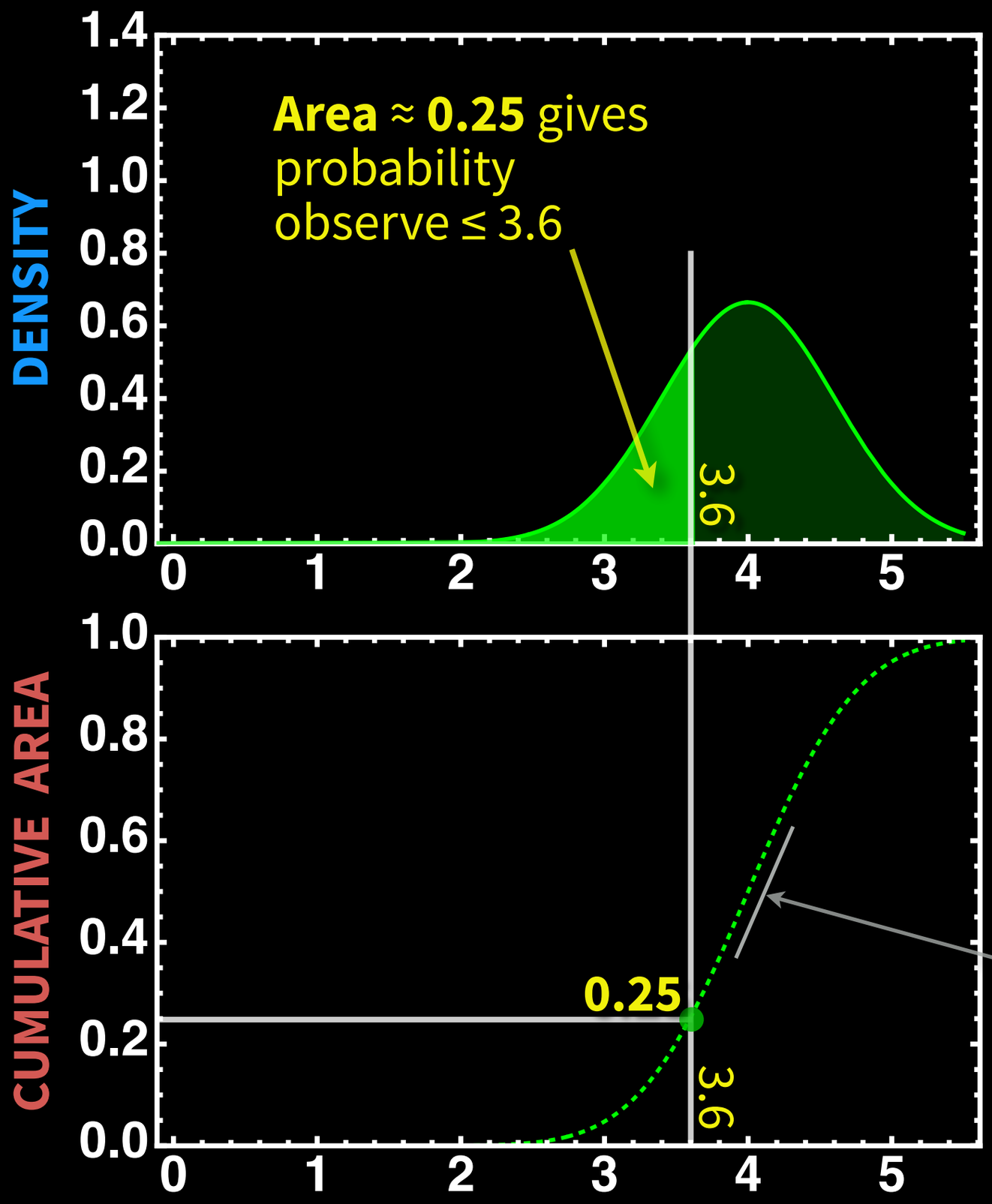
distribution of control

distribution of treated

PROBABILITY DENSITY FUNCTION (pdf) is the curve; **PROBABILITY = AREA UNDER**

*Curve higher = more likely to observe near there… **modes** are where the maxima are*

Cumulative View of Distributions



Probability density function

PDF $f(x) = F(x)'$ = likelihood of observation infinitesimally close to x

same information

Cumulative density function

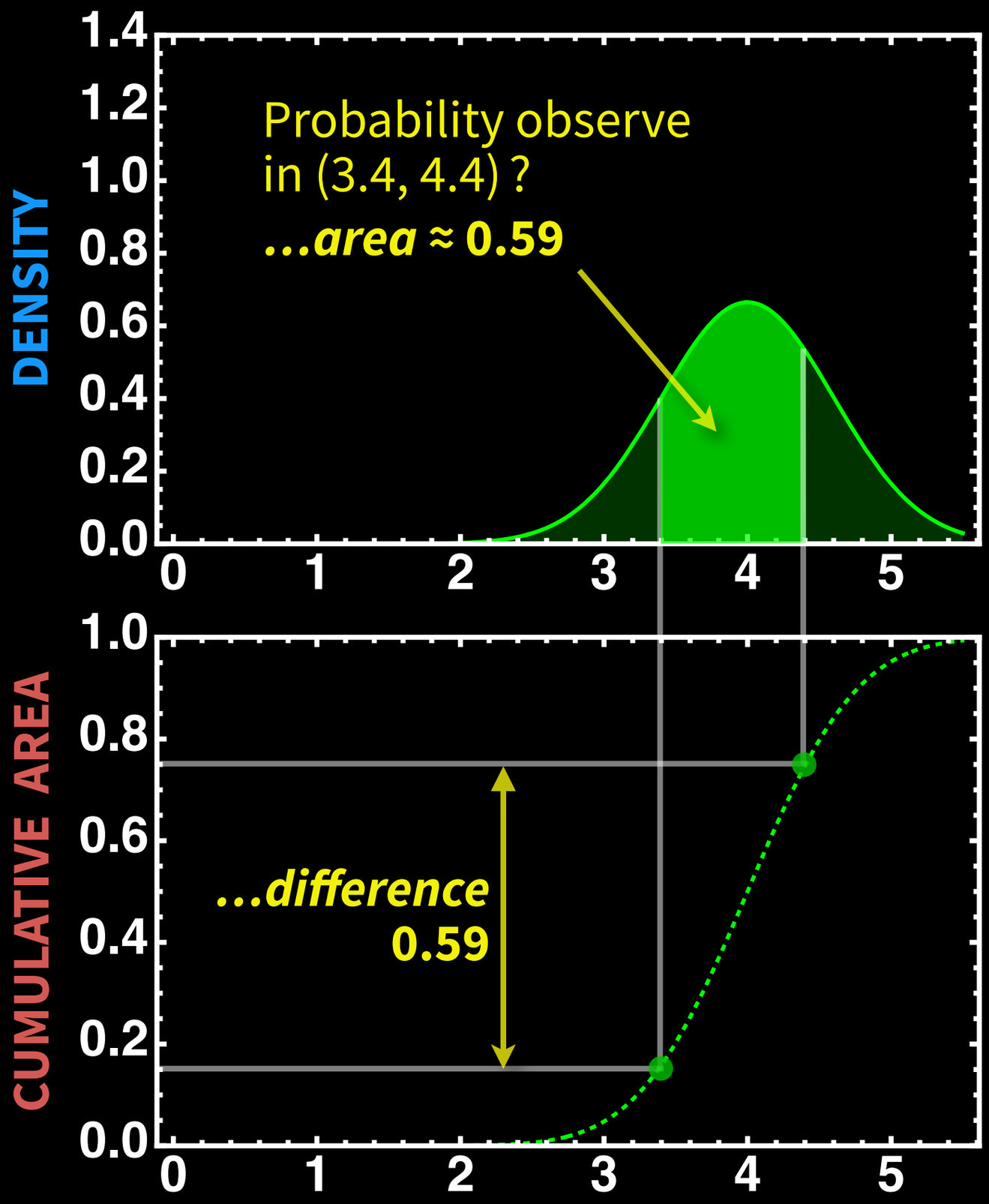
CDF $F(x) = \int_{-\infty}^{x} f(y)\,dy =$ Probability[observe ≤ x]

observations likely to be found where steeply sloped

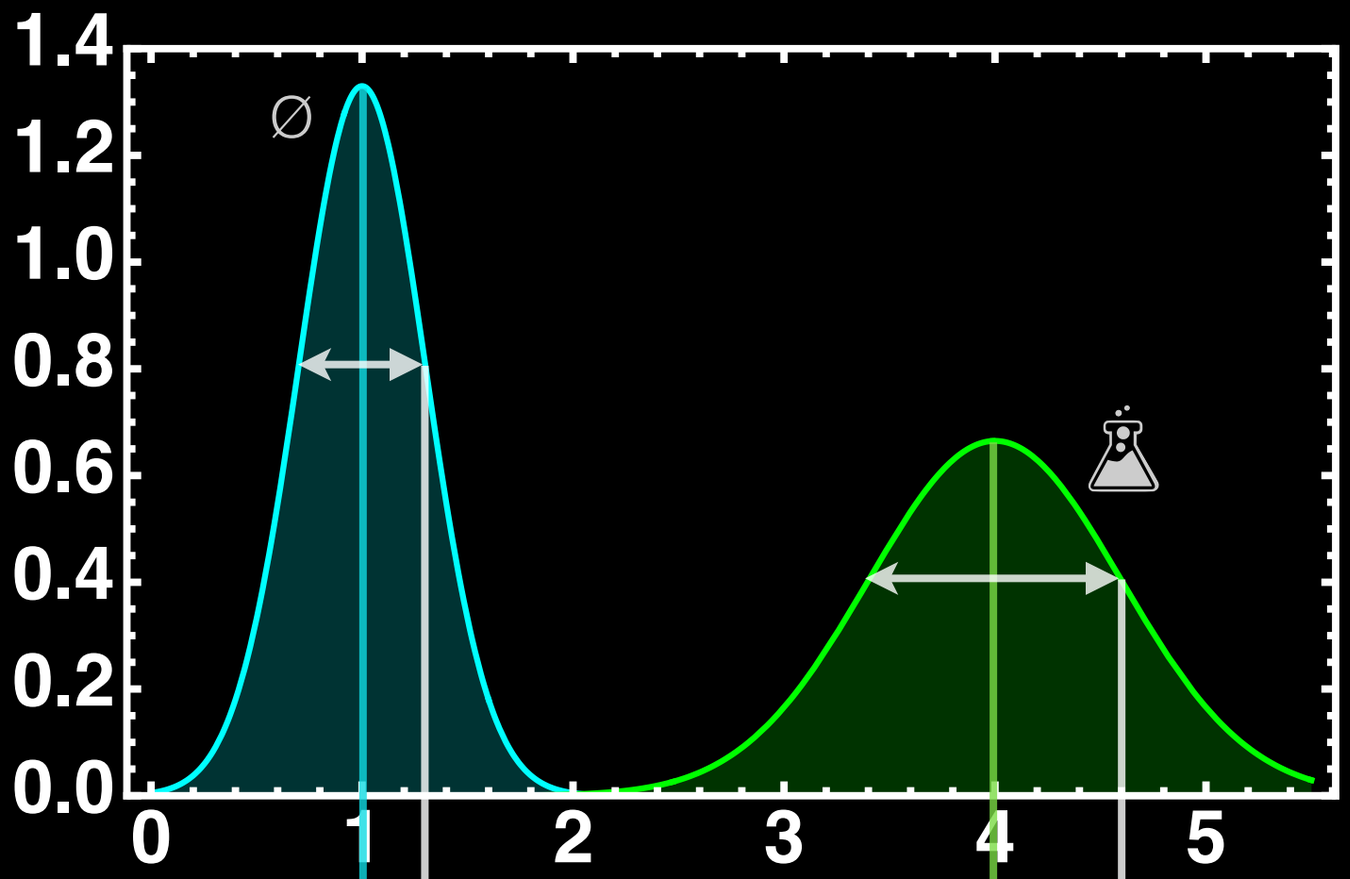
If do cumulative from the right (Pr > x) then get "survival function"

SF = 1 – CDF

Cumulative View of Distributions



Descriptive Stats. of Distributions



Infinity observations per group per experiment, bins of width zero

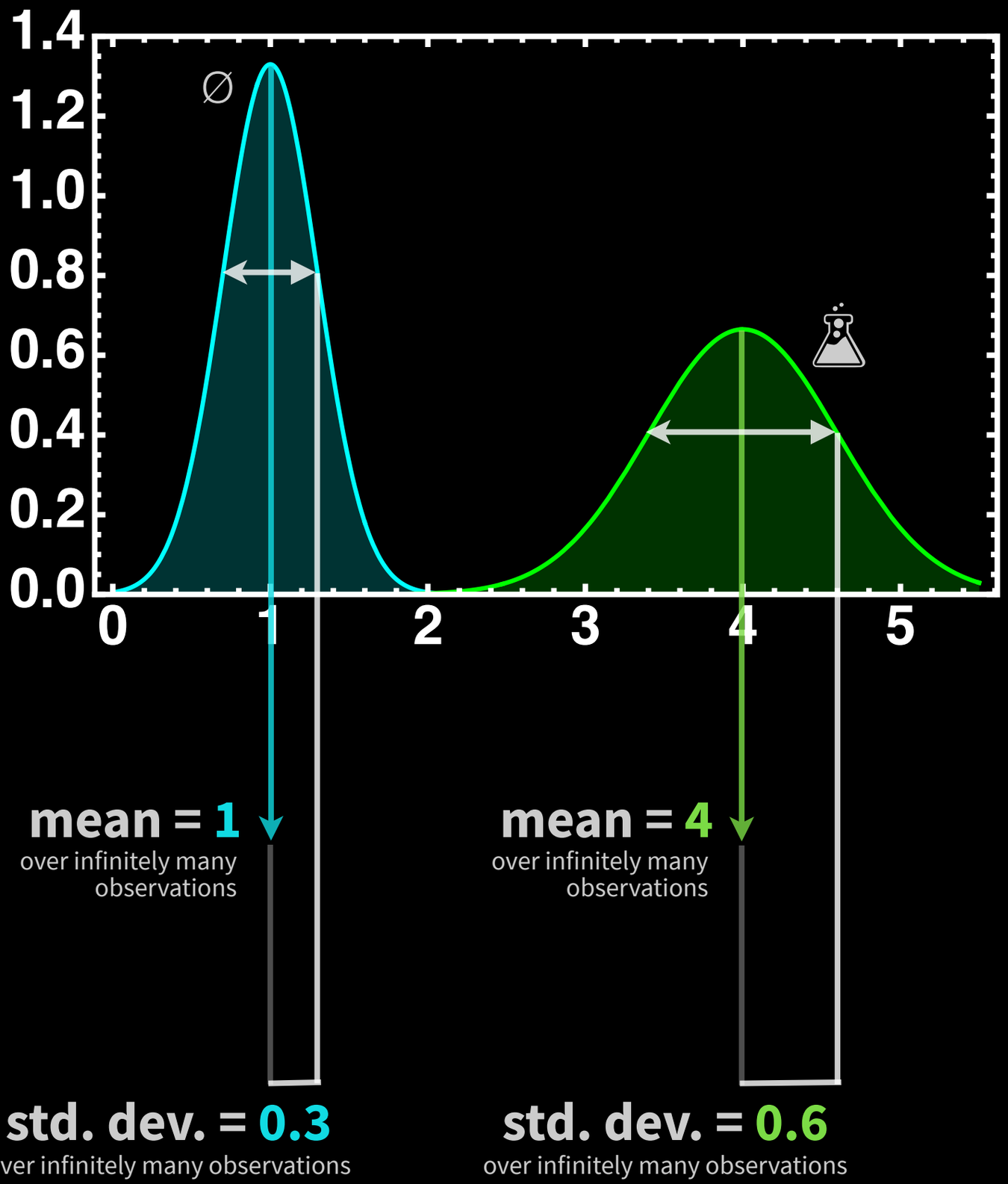
mean = 1
over infinitely many observations

mean = 4
over infinitely many observations

std. dev. = 0.3
over infinitely many observations

std. dev. = 0.6
over infinitely many observations

Descriptive Stats. of Distributions

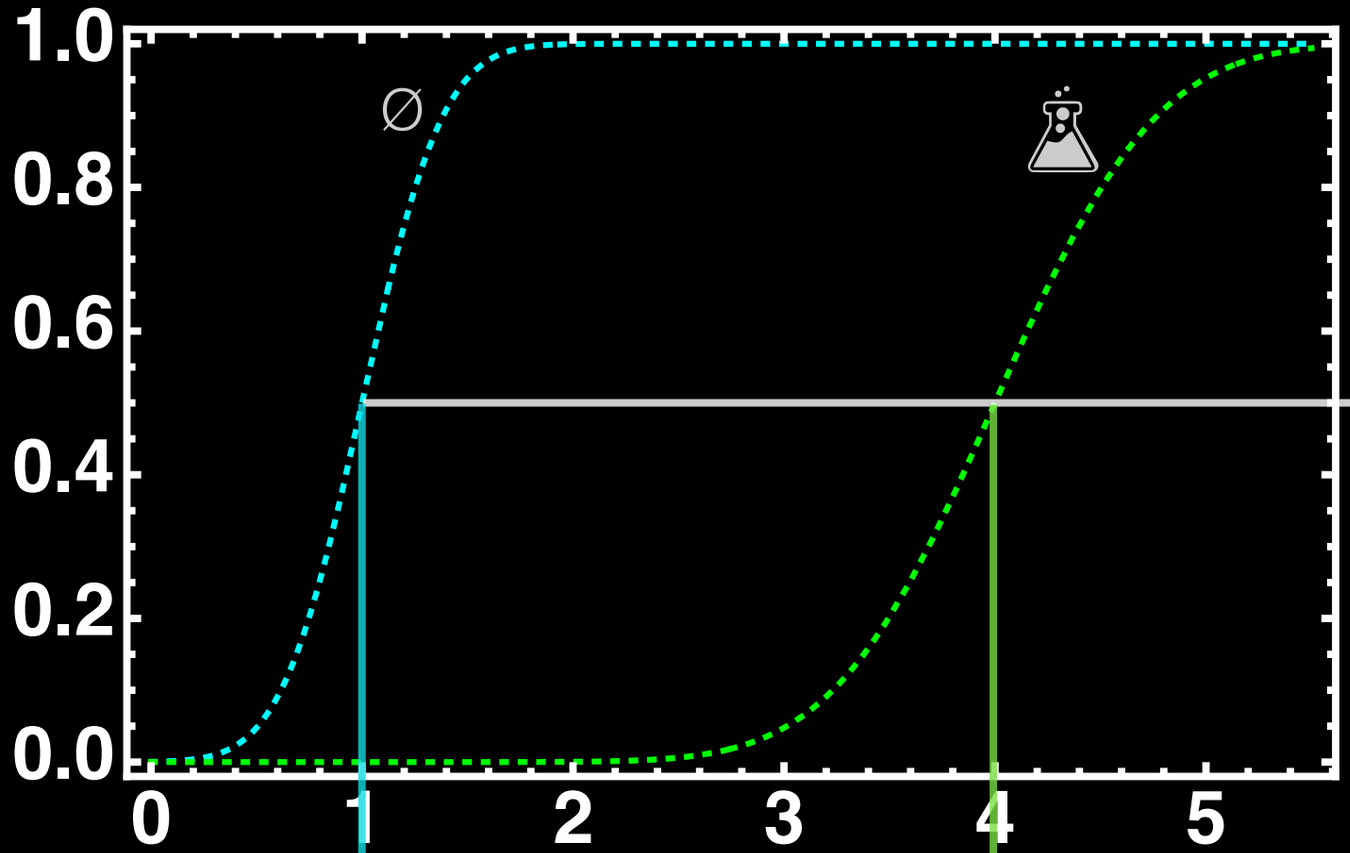


Infinity observations per group per experiment, bins of width zero

Some distributions are too "broad" to have a finite or defined variance/standard deviation, or even a mean.

For example, a soccer penalty kick in a uniformly random angle toward the endline (on an infinitely wide field) crosses the endline in a Cauchy distribution, which has undefined mean.

Descriptive Stats. of Distributions



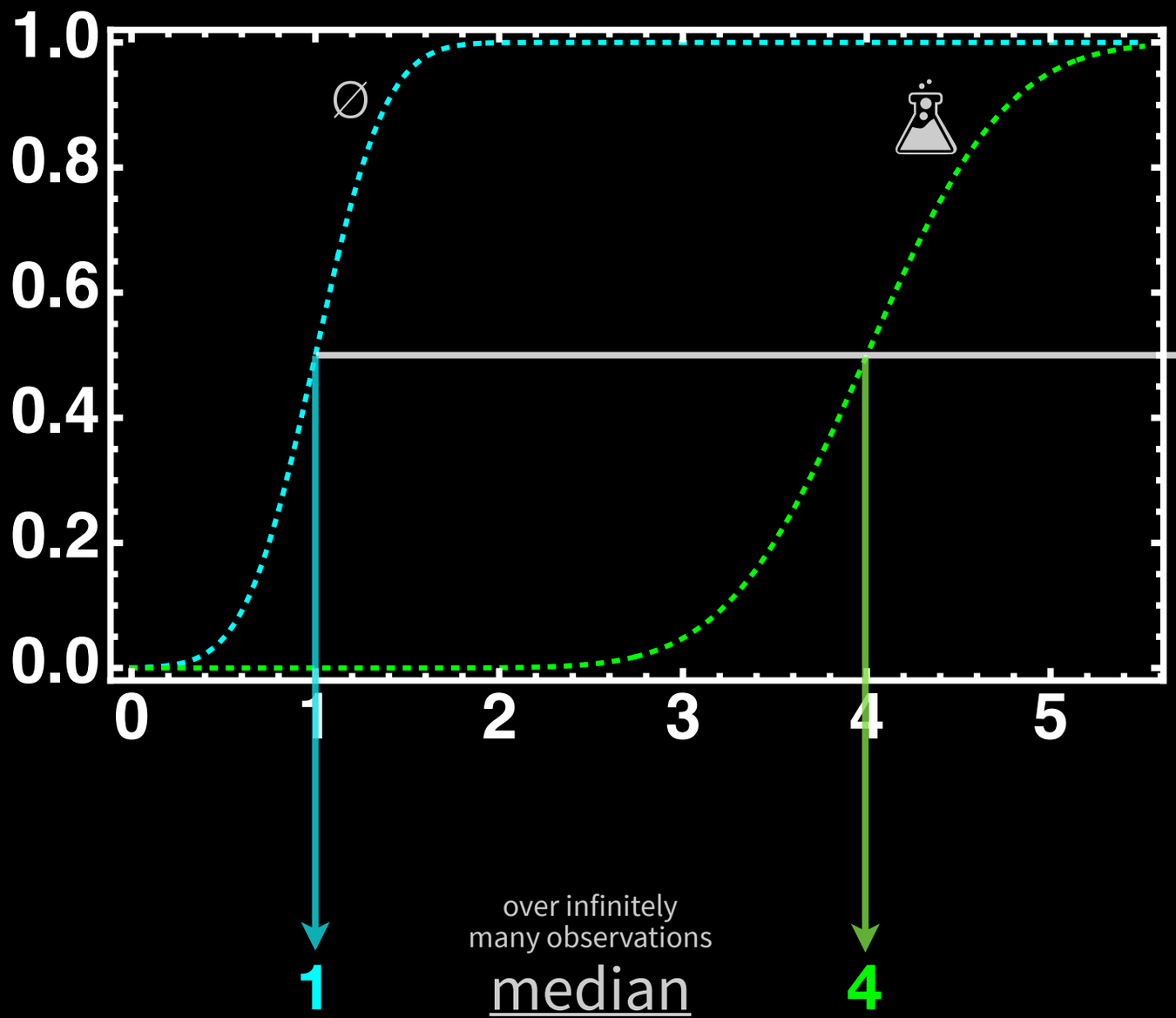
over infinitely
many observations

1 median **4**

is levels where half observations are below
and half of observations are above

R function: `median(…)`

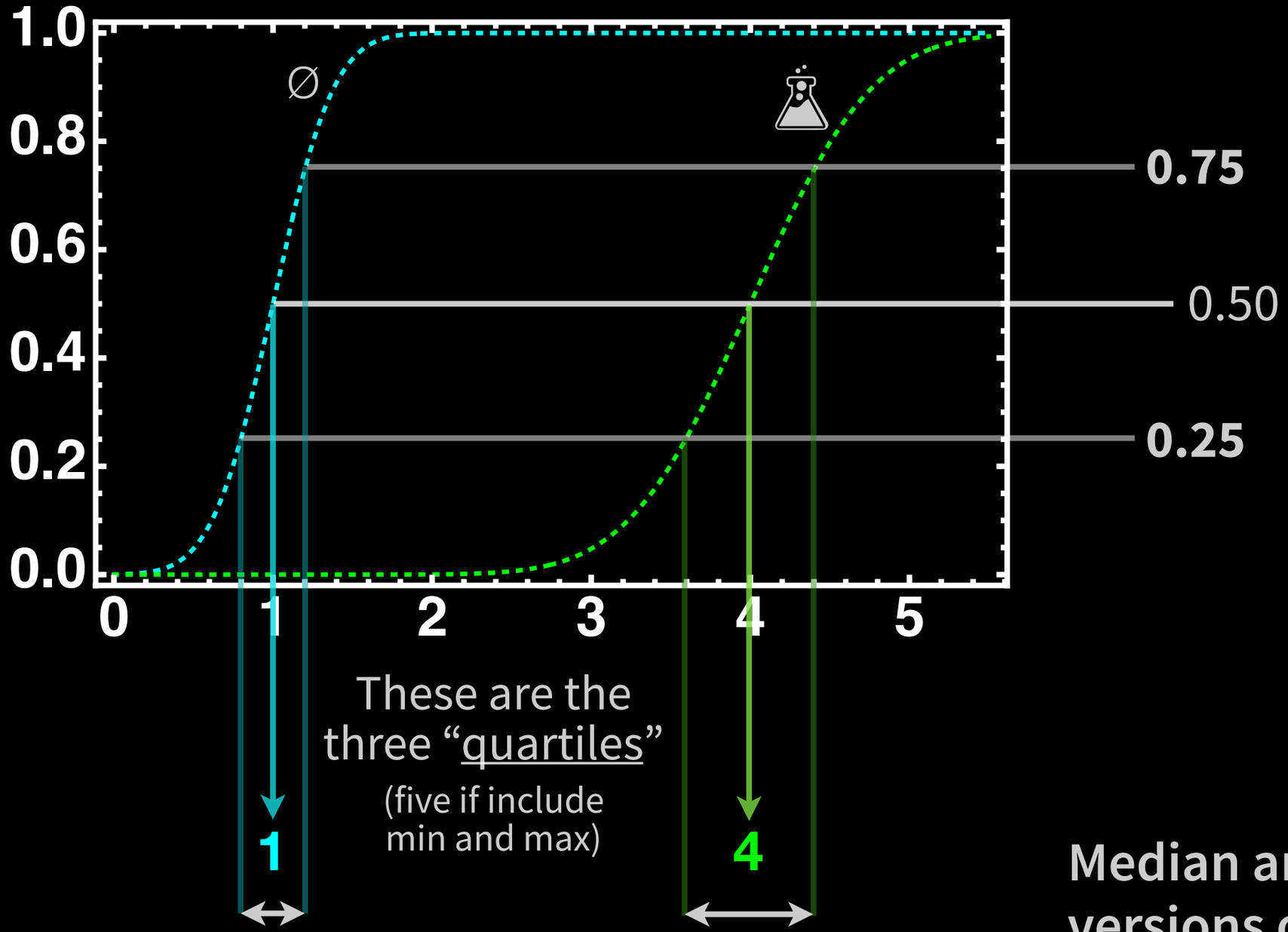
Descriptive Stats. of Distributions



is levels where half observations are below and half of observations are above

There are lots of measures of "central tendency" / position / center: (arithmetic) mean, geometric mean, harmonic mean, power means, median, modes, weighted means, trimmed means, …

Descriptive Stats. of Distributions



IQR: "Inter-Quartile Range"

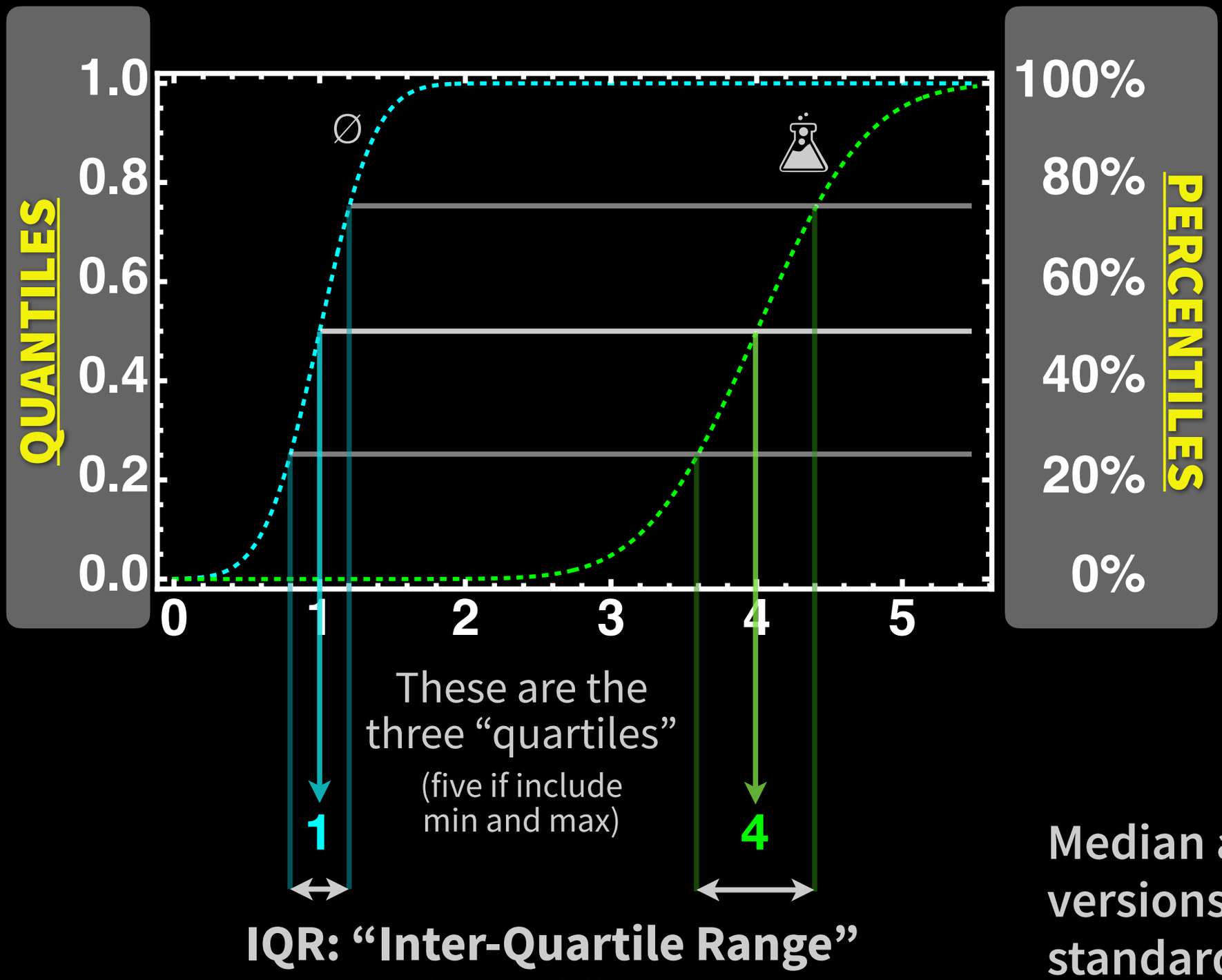
…the "middle 50%"

Median and IQR are "robust" versions of mean and (scaled) **standard dev.** *…more later…*

…can also estimate them from finite-sized samples.

R functions: `quantile(…)` `IQR(…)`

Descriptive Stats. of Distributions



These are the three “quartiles”

(five if include min and max)

IQR: “Inter-Quartile Range”

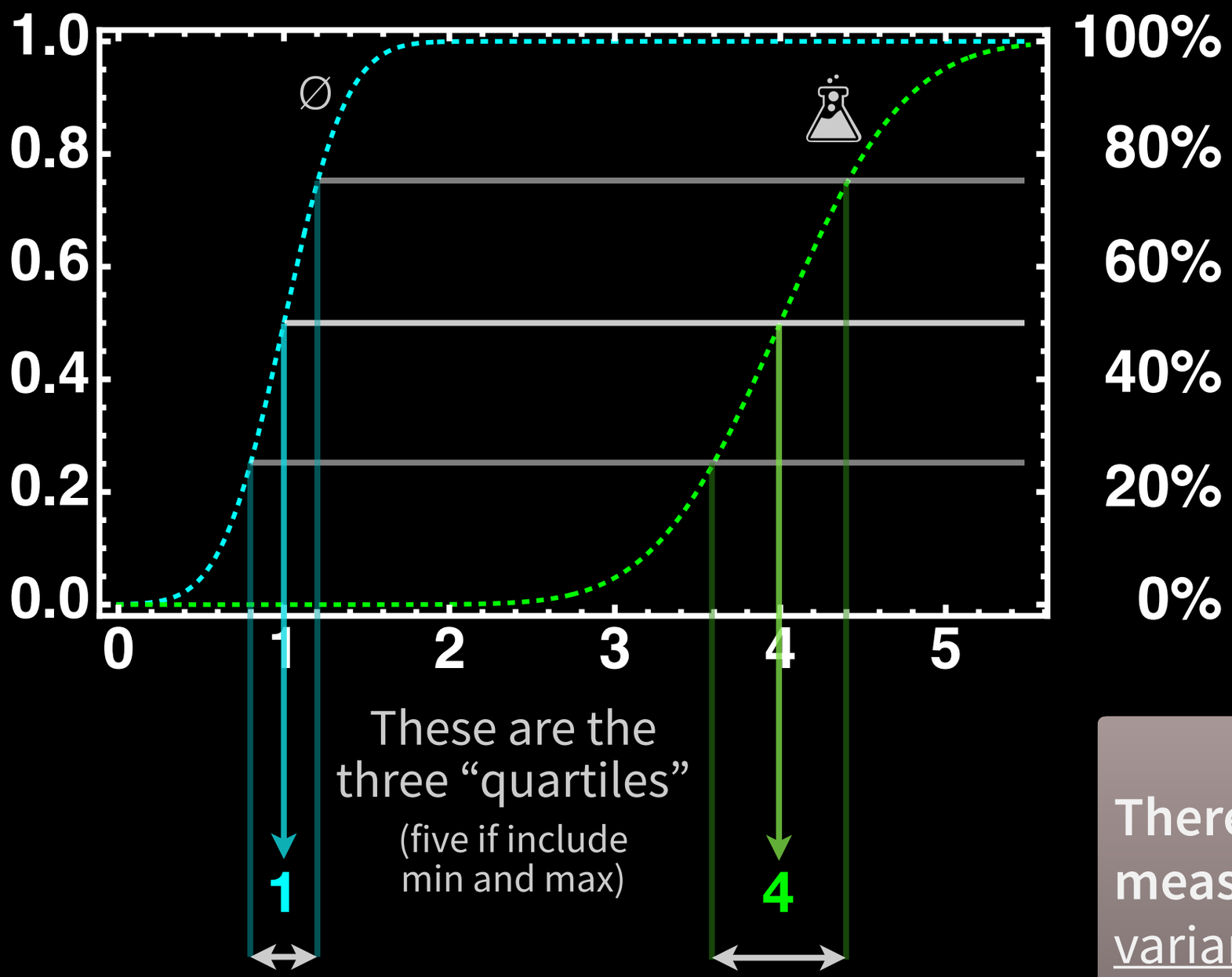
…the “middle 50%”

Median and IQR are “robust” versions of mean and (scaled) **standard dev.** *…more later…*

…can also estimate them from finite-sized samples.

R functions: `quantile(…)` `IQR(…)`

Descriptive Stats. of Distributions



There are also lots of measures of “dispersion”:
variance, standard deviation, mean absolute deviation (MAD), range, quantiles/percentiles, IQR, coefficient of variation, …

R functions: `quantile(…)` `IQR(…)`

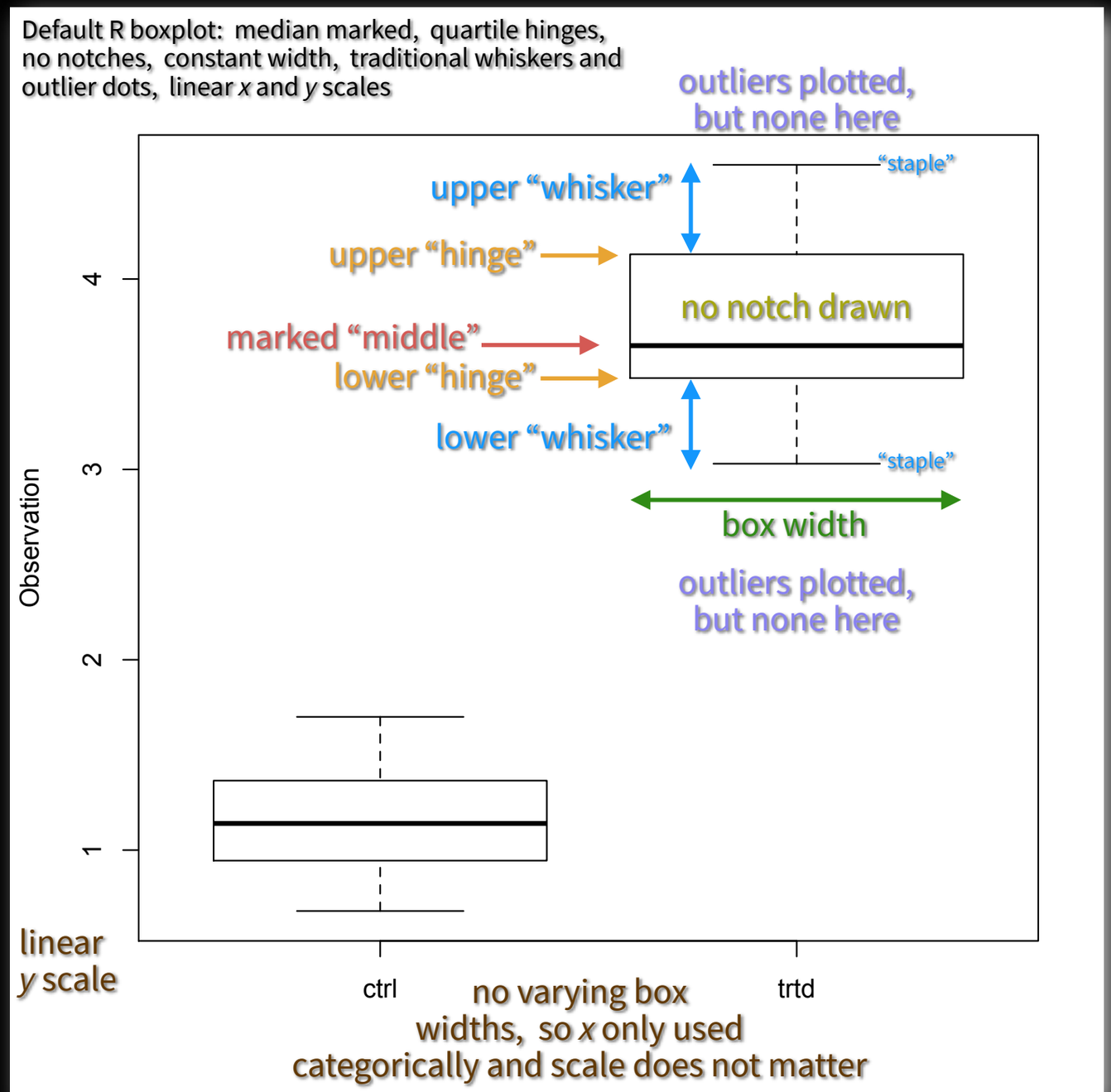
Box Plots

```
?boxplot
?bxp
?boxplot.stats
```

```
YLAB <- "Observation";
boxplot(obs ~ grp, data=df, ylab=YLAB);   # …data.frame  and formula input
boxplot(ctrl, trtd, names=c("ctrl", "trtd"), ylab=YLAB);  # …vector inputs
```

Box(-and-whisker) **plots have many variations:**

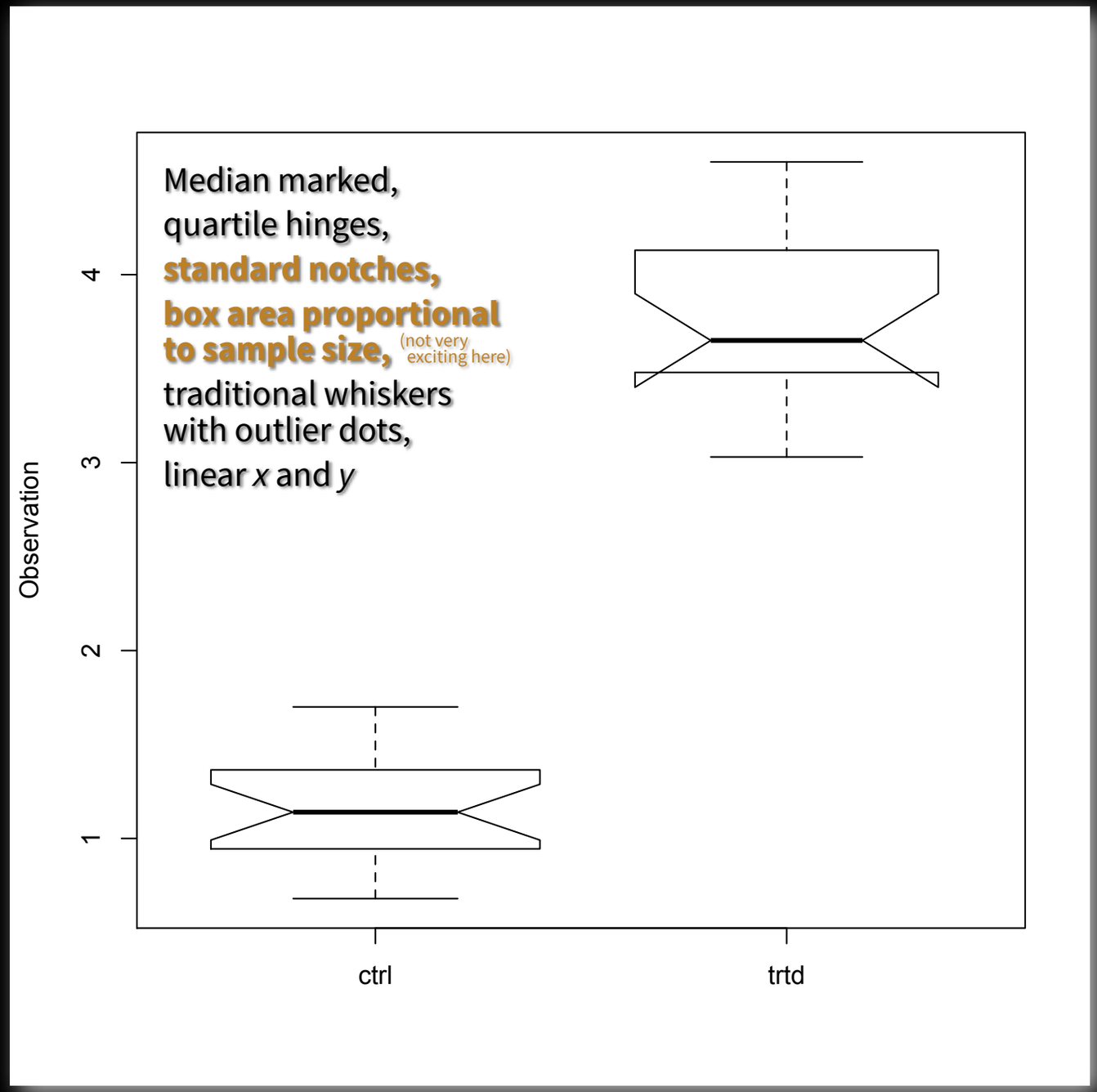
- **Marked middle** of box usually shows **median** *but sometimes mean is (also) shown.*
- **Lower/upper limits of box** ("hinges") are usually estimates of **0.25 and 0.75 quantile.**
- **Notched boxes usually show some kind of confidence interval for median differences**, e.g., with ± 1.58 · IQR / √sampleSize then two notches almost overlapping is ≈ 0.05 level for significant difference of medians *(if sample sizes are similar and not too small from continuous distributions, because then have asymptotic normality of medians).*
- **Width of the box** may be constant, or related to **sample size** *(e.g., proportional to box area).*
- **Whiskers vary:** traditionally, they go out to *furthest **datapoints** that are* ≤ 1.5 · box width *further out from hinges*; sometimes: min, max datapoint. **Datapoints beyond the whiskers ("outliers") may or may not be shown.** Large normal-dist. samples: expected to have whiskers at ≈ ±2.7σ, ≈99.3% within whiskers.
- *x* and/or *y* axes may be log scale. **DOCUMENT!**



Box Plots

```
?boxplot
?bxp
?boxplot.stats
```

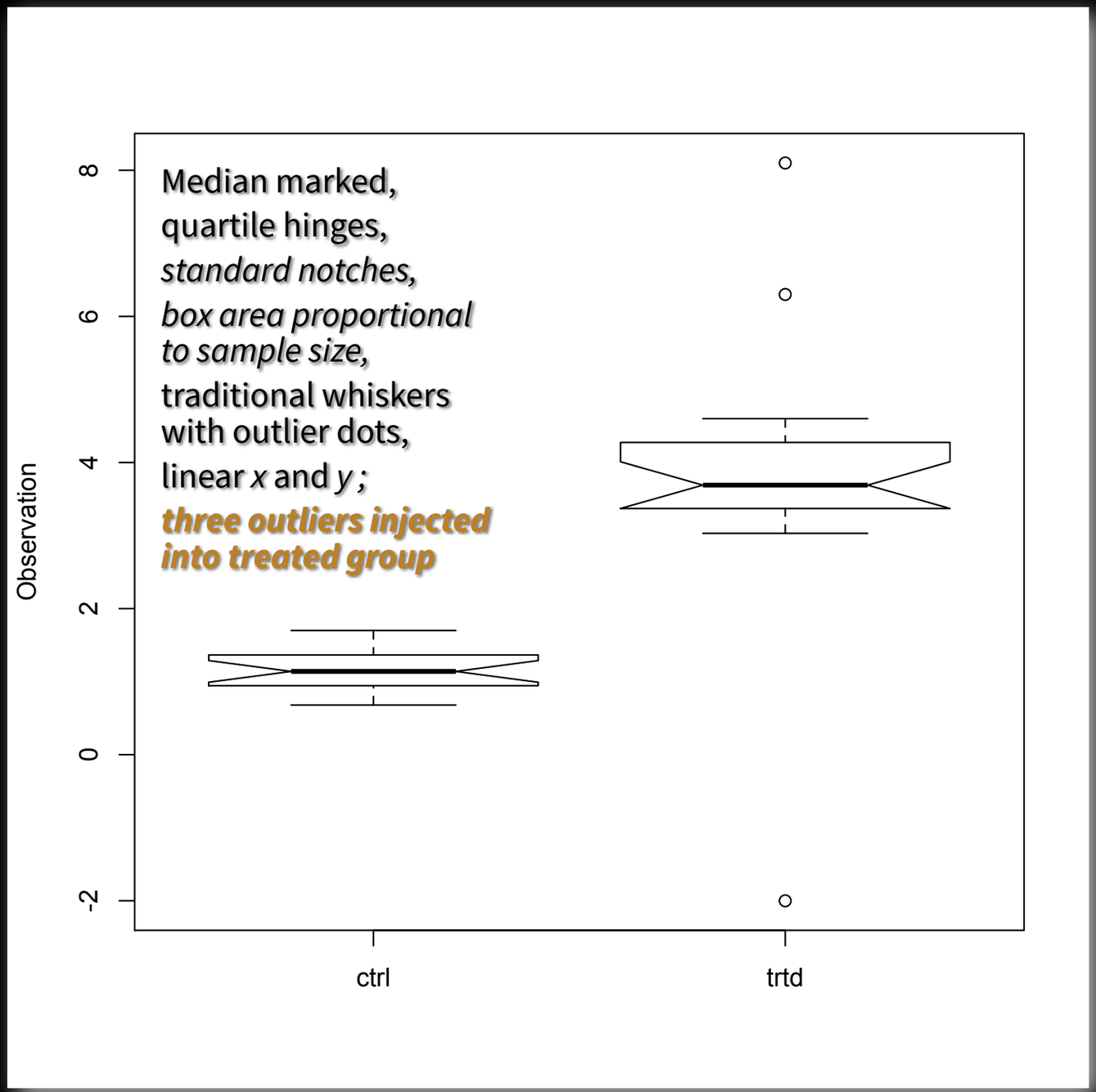
```
YLAB <- "Observation";
boxplot(obs ~ grp, data=df, ylab=YLAB);
boxplot(ctrl, trtd, names=c("ctrl", "trtd"), ylab=YLAB);
boxplot(obs ~ grp, data=df,
  varwidth=TRUE, notch=TRUE,
  ylab=YLAB);
```



Box Plots

```
?boxplot
?bxp
?boxplot.stats
```

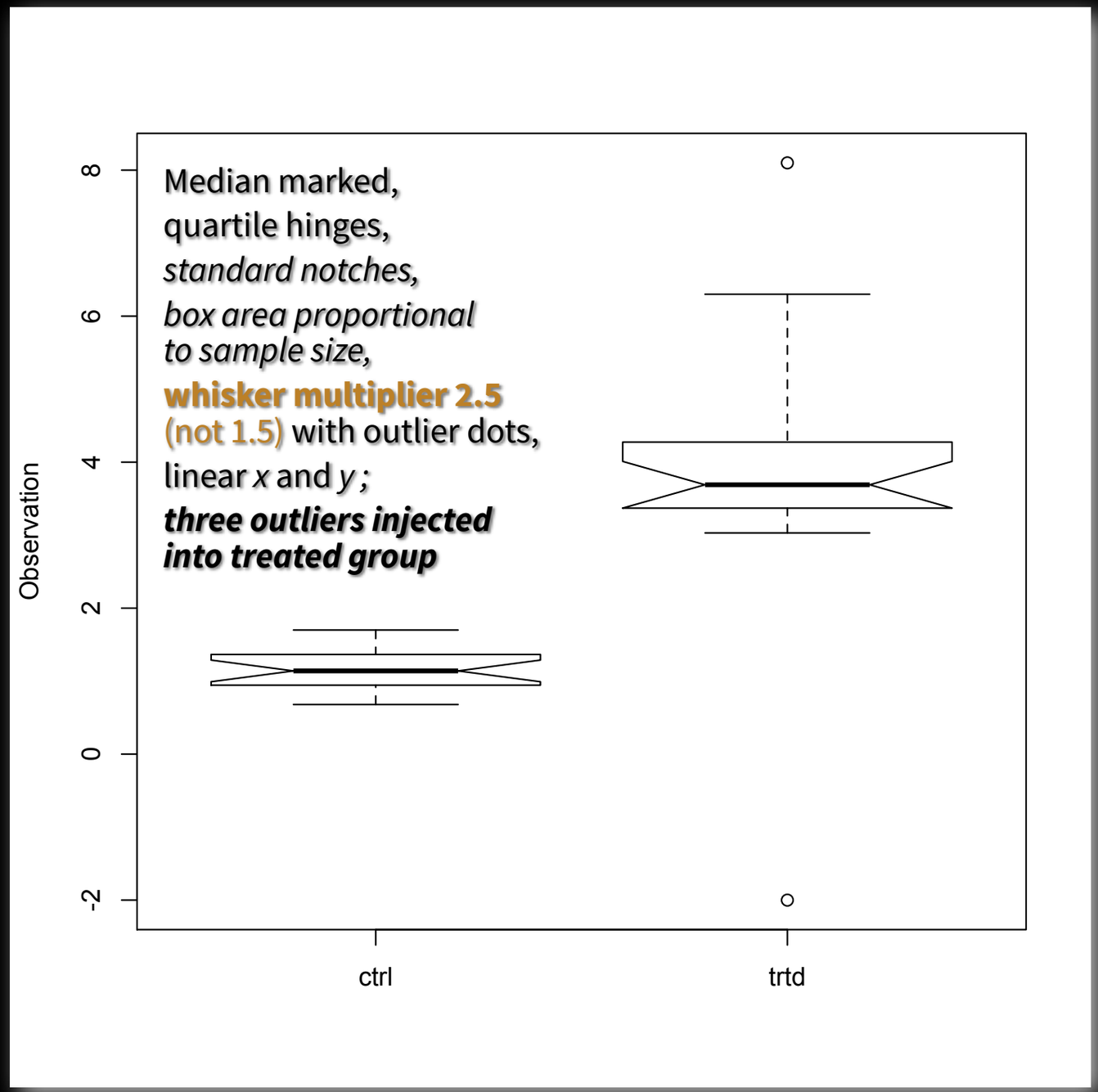
```
YLAB <- "Observation";
boxplot(obs ~ grp, data=df, ylab=YLAB);
boxplot(ctrl, trtd, names=c("ctrl", "trtd"), ylab=YLAB);
boxplot(obs ~ grp, data=df,
  varwidth=TRUE, notch=TRUE,
  ylab=YLAB);
boxplot(ctrl,
  c(trtd, -2.0, 6.3, 8.1),
  names=c("ctrl", "trtd"),
  varwidth=TRUE, notch=TRUE,
  ylab=YLAB);
```



Box Plots

```
?boxplot
?bxp
?boxplot.stats
```

```
YLAB <- "Observation";
boxplot(obs ~ grp, data=df, ylab=YLAB);
boxplot(ctrl, trtd, names=c("ctrl", "trtd"), ylab=YLAB);
boxplot(obs ~ grp, data=df,
  varwidth=TRUE, notch=TRUE,
  ylab=YLAB);
boxplot(ctrl,
  c(trtd, -2.0, 6.3, 8.1),
  names=c("ctrl", "trtd"),
  varwidth=TRUE, notch=TRUE,
  ylab=YLAB);
boxplot(ctrl,
  c(trtd, -2.0, 6.3, 8.1),
  names=c("ctrl", "trtd"),
  varwidth=TRUE, notch=TRUE,
  ylab=YLAB, range=2.5);
```

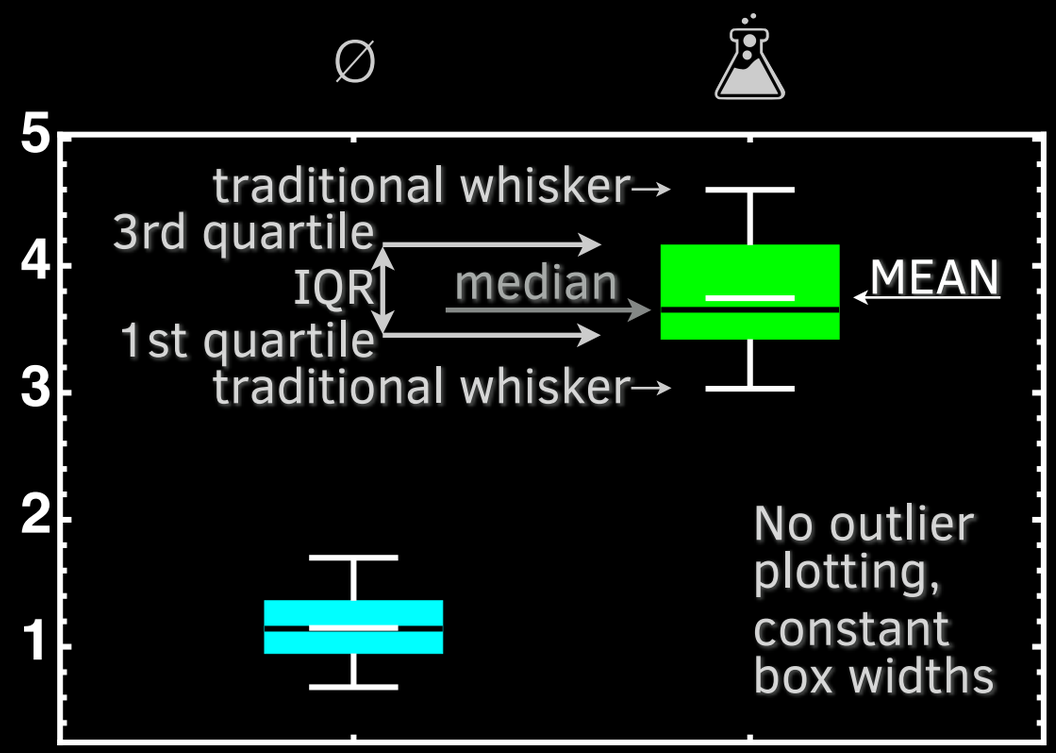


Continue Our Experimental Example:

∅

1.22	0.73	1.06	0.73	1.16
0.77	1.11	1.47	1.70	0.91
1.32	1.51	1.12	0.68	1.10
1.41	0.98	1.24	1.46	1.27

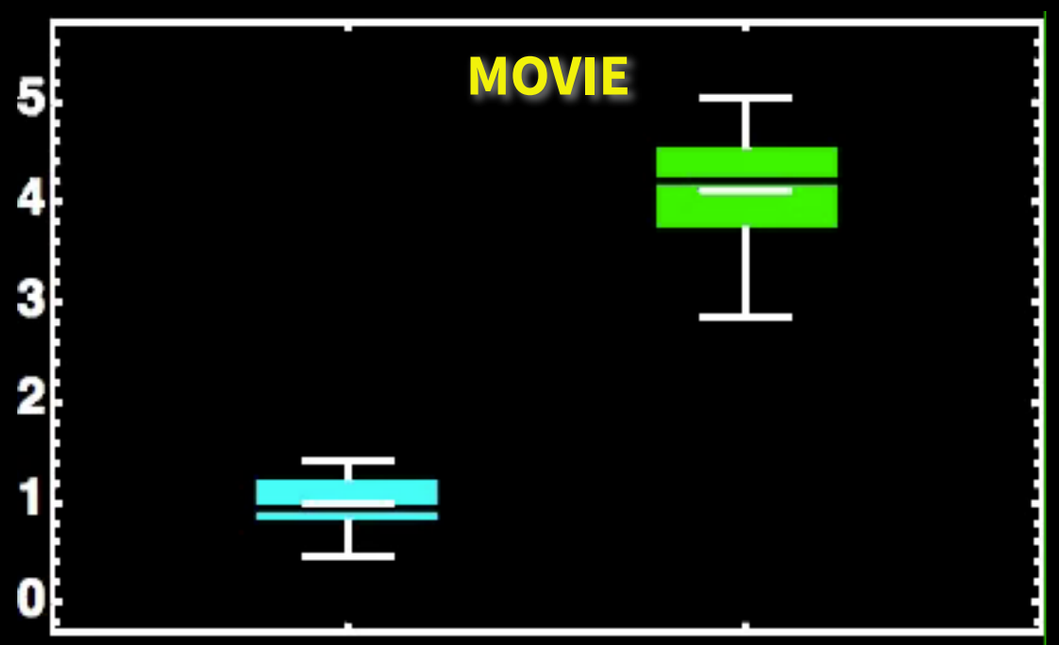
4.60	4.39	4.29	3.55	3.26
4.13	3.21	4.26	3.99	3.53
3.73	3.19	3.49	3.65	3.48
3.03	3.88	3.97	4.16	4.29



Continue Our Experimental Example:

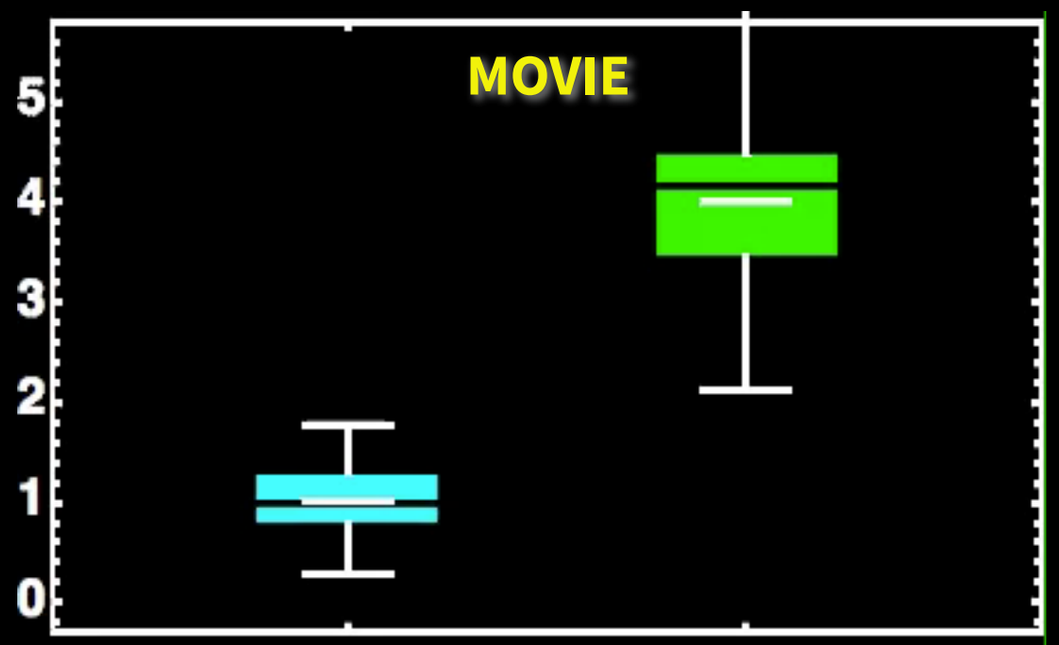
y axes are kept locked, even though some experiments have boxplots that go outside *y* limits ; don't worry much about this

20 and **17** replicates per experiment:



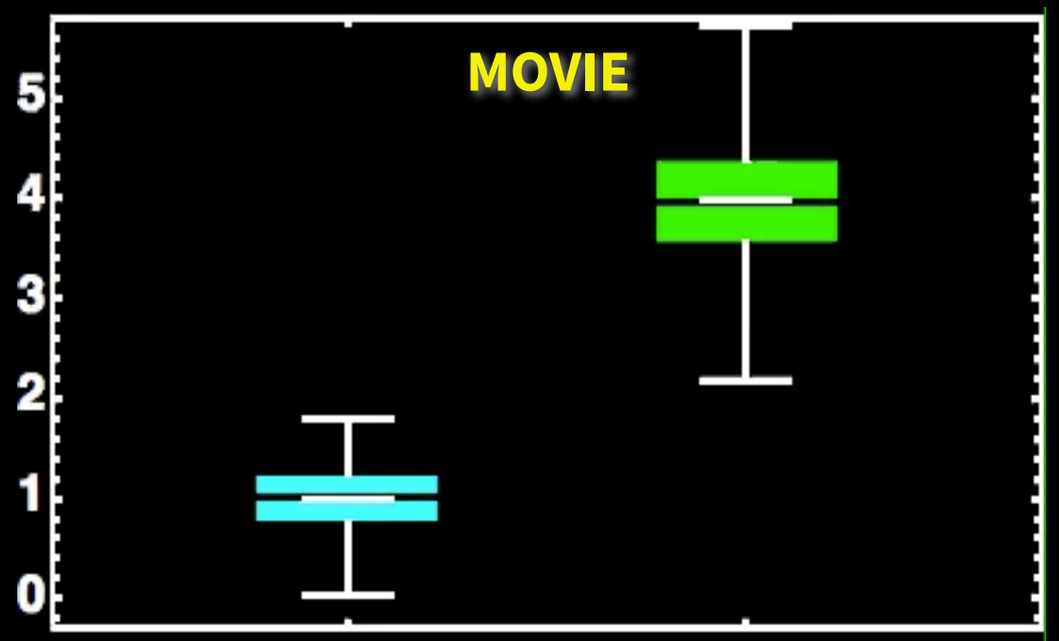
Continue Our Experimental Example:

100 and **100** replicates per experiment:



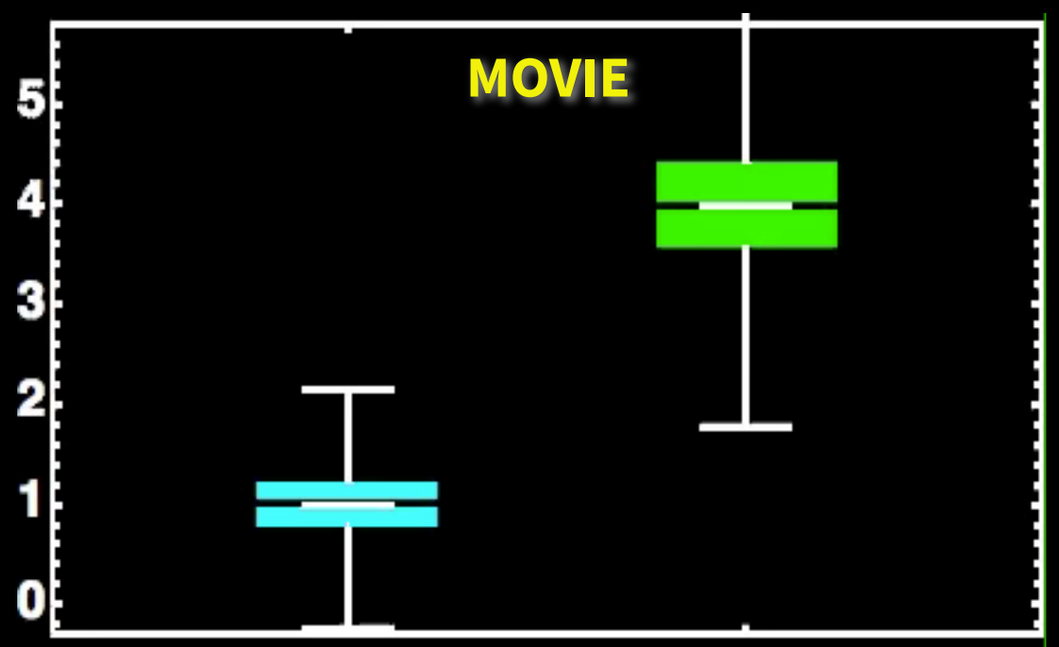
Continue Our Experimental Example:

1,000 and 1,000 replicates per experiment:



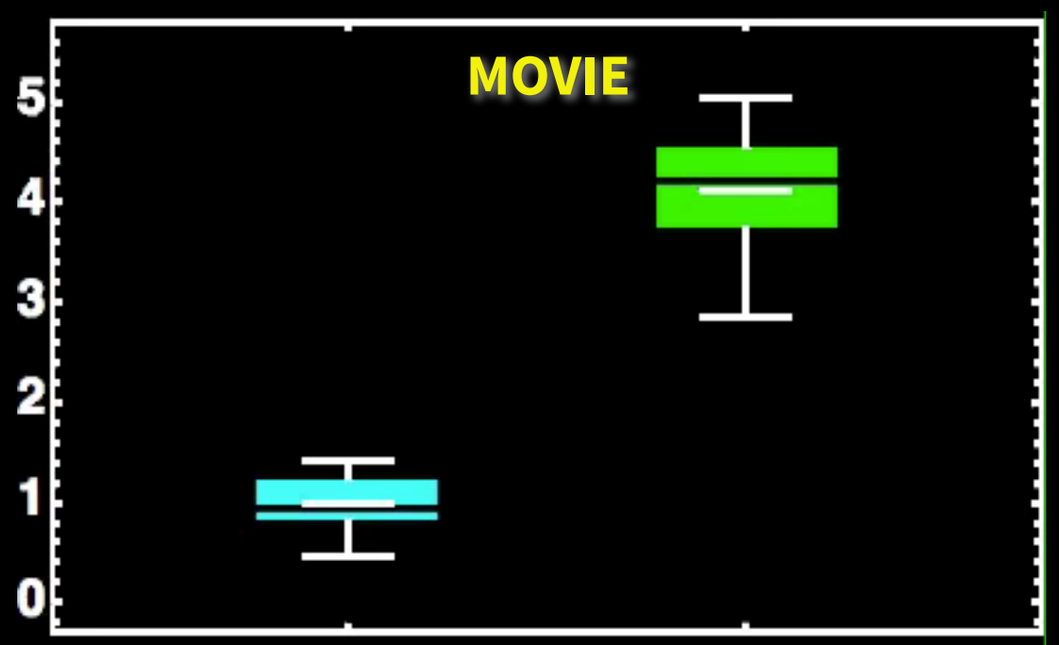
Continue Our Experimental Example:

10,000 and **10,000** replicates per experiment:



Continue Our Experimental Example:

20 and **17** replicates per experiment:



Many Distributions and Named Families!

arcsine • Bates • Benford • Benini • Benktander-Gibrat • **Bernoulli** • **beta** (regular, non-central) • beta-binomial • beta-negative binomial • beta-prime • **binomial** • Birnbaum-Saunders • Borel-Tanner • Burr • Cauchy • censored … • **chi** • **chi-square** (regular, non-central) • copula … • Coxian • Dagum • David • Dirichlet • distributions from graphs • distributions from matrix properties • **empirical** (including kernel-smoothed) • Erlang • exp-gamma • **exponential** • exponential-power • extreme value • ***F* ratio** (regular, non-central) • failure • Fisher *z* • Frechet • **gamma** • **geometric** • Gumbel • half-normal • Hotelling *T*-square • hyperbolic • hyperbolic secant • hyperexponential • **hypergeometric** (regular, Wallenius, Fisher, multivariate) • hypoexponential • inverse chi-square • inverse gamma • inverse gaussian • Johnson • Kumaraswamy • Landau • Laplace • Levy • Lindley • log-gamma • log-logistic • log-series • logistic • **lognormal** (regular, multivariate) • Marchenko-Pastur • marginal … • matrix circular (real, unitary, quaternion, orthogonal, symplectic) • matrix Gaussian (regular, orthogonal, unitary, symplectic) • matrix inverse Wishart • matrix normal • matrix *t* • matrix Wishart • max-stable • Maxwell • Meixner • min-stable • mixture … • Moyal • **multinomial** • Nakagami • **negative binomial**/multinomial • **NORMAL** (1-D, binormal/multinormal) • order … • Pareto • Pareto-Pickands • Pascal • Pearson • PERT • piecewise/spliced … • **Poisson** (1-D, multivariate) • Poisson-Consul • Polya-Aeppli • power • process slice … • product … • Rayleigh • reliability • Rician • shifted Gompertz • sign-rank • Singh-Maddala • Skellam • skew-normal • stable • standby • stationary • **Student *t*** (regular, non-central, multivariate) • survival • Suzuki • Tracy-Widom • transformed … • triangular • truncated … • Tsallis *q*-exponential • Tsallis *q*-Gaussian • Tukey lambda • **uniform** (1-D/multivariate discrete/continuous) • uniform sum • variance-gamma • Voigt • von Mises • Wakeby • Waring-Yule • Weibull • Wigner semicircle • Wilcox • Zipf • …

Example: Continuous UNIFORM

Parameters: endpoints… two real numbers $a < b$

Support: real interval (a, b)

Definition: never in $(-\infty, a]$ or $[b, \infty)$; otherwise: constant density, i.e., prob. is proportional to width of interval for intervals within (a, b)

PDF at x: $1/(b-a)$ if $a < x < b$, else 0 — *R*: `dunif(…)`

CDF at x: $(x-a)/(b-a)$ if $a < x < b$, 0 if $x \le a$, 1 if $x \ge b$ — *R:* `punif(…)`

Mean: $(a+b)/2$ — *R:* `runif(…)` *simulates*

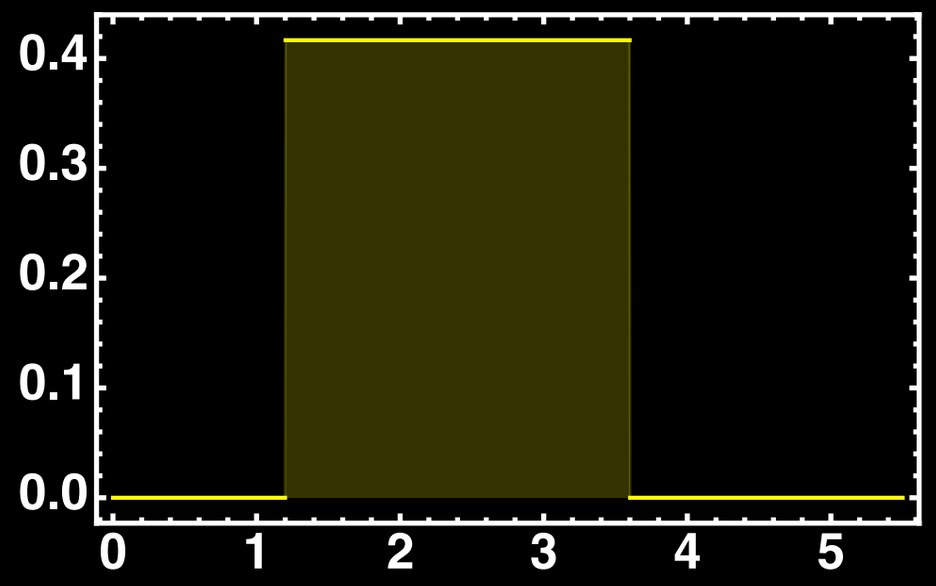
Variance: $(b-a)^2/12$

Quartiles: $(3a+b)/4$, $(a+b)/2$, $(a+3b)/4$ — *R:* `qunif(…)`

Modes: entire interval (a, b)

Comments: often what people mean when they say "at random" without further qualification (if a and b are clear)

PDF of uniform on (1.2, 3.6)**:**

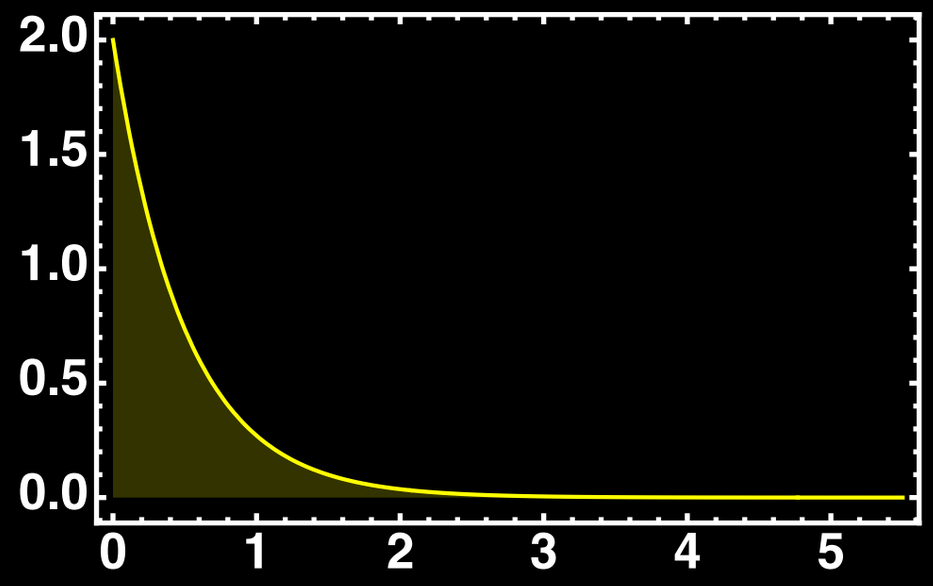


Example: EXPONENTIAL

AUDIO

Parameters:	rate… real $\lambda > 0$	
Support:	real interval $(0, \infty)$	
Definition:	distribution of waiting times between events if they occur at a constant rate of λ events per unit time	
PDF at x:	$\lambda/(e^{\lambda x})$ if $x > 0$, else 0	*R*: `dexp(…)`
CDF at x:	$1 - e^{-\lambda x}$ if $x > 0$, else 0	*R:* `pexp(…)`
Mean:	$1/\lambda$	*R:* `rexp(…)` *simulates*
Std. deviation:	$1/\lambda$	
Quartiles:	$\ln(4/3)/\lambda \approx 0.29/\lambda$, $\ln(2)/\lambda \approx 0.69/\lambda$, $\ln(4)/\lambda \approx 1.39/\lambda$	*R:* `qexp(…)`
Modes:	0 (PDF strictly decreases after 0)	
Example:	time between Geiger counter clicks in front of a constant radiation source	

PDF of exponential 2.0**:**



Example: GAMMA

Parameters: number of events, time per event… reals* $\alpha, \beta > 0$

Support: real interval $(0, \infty)$

Definition: distribution of waiting time for α events to occur if they occur at a constant rate of $1/\beta$ events per unit time, i.e., *sum of α independent exponental $1/\beta$'s*

PDF at x: $x^{\alpha-1} / (\beta^{\alpha} \cdot \Gamma(\alpha) \cdot e^{x/\beta})$ if $x > 0$, else 0 — *R:* `dgamma(…)`

(gamma special function generalizes factorial)

CDF at x: *« regularized incomplete gamma special function »* — *R:* `pgamma(…)`

Mean: $\alpha \cdot \beta$ — *R:* `rgamma(…)` *simulates*

Variance: $\alpha \cdot \beta^2$

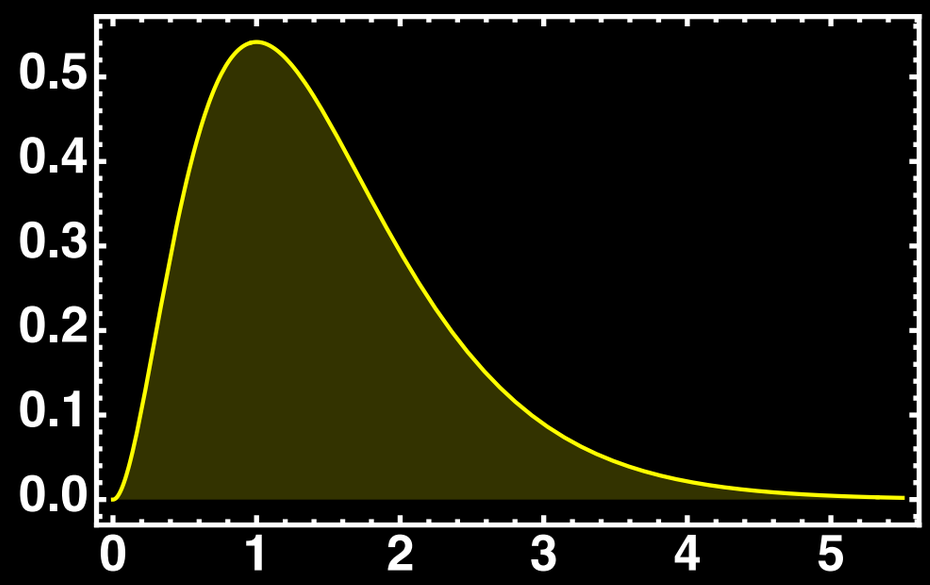
Quartiles: *« …inverse regularized incomplete gamma… »* — *R:* `qgamma(…)`

Modes: $(\alpha-1) \cdot \beta$ if $\alpha > 1$, else 0

Comments: $\alpha = 1$ case same as exponential $\lambda = 1/\beta$

**Like many distributions, alternative parameterizations exist and are in use, so you need to figure out which one is used in a piece of software or literature!*

PDF of gamma 3.0, 0.5**:**



Example: GAMMA

Parameters: number of events, time per event… reals* $\alpha, \beta > 0$

Support: real interval $(0, \infty)$

Definition: distribution of waiting time for α events to occur if they occur at a constant rate of $1/\beta$ events per unit time, i.e., *sum of α independent exponental $1/\beta$'s*

PDF at x: $x^{\alpha-1} / (\beta^{\alpha} \cdot \Gamma(\alpha) \cdot e^{x/\beta})$ if $x > 0$, else 0 — *R:* `dgamma(…)`

(gamma special function generalizes factorial)

CDF at x: *« regularized incomplete gamma special function »* — *R:* `pgamma(…)`

Mean: $\alpha \cdot \beta$ — *R:* `rgamma(…)` *simulates*

Variance: $\alpha \cdot \beta^2$

Quartiles: *« …inverse regularized incomplete gamma… »* — *R:* `qgamma(…)`

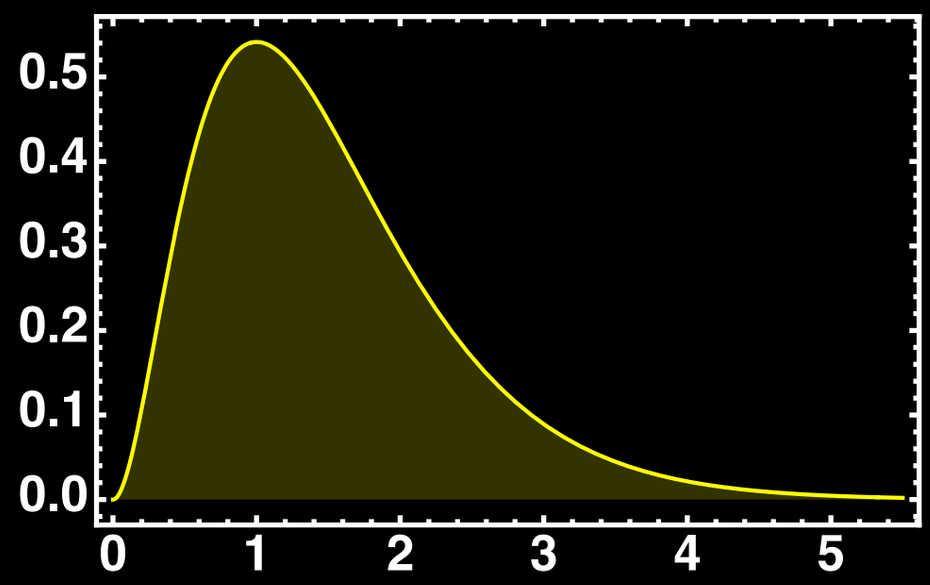
Modes: $(\alpha - 1) \cdot \beta$ if $\alpha > 1$, else 0

Comments: $\alpha = 1$ case same as exponential, $\lambda = 1/\beta$

Like many distributions, alternative parameterizations exist, so you need to figure out which one is used in a piece of software or literature.

There are descriptive statistics that measure "skewness" — stretched out left tail ("negative skew" with probability mass concentrated on the right) **vs. case here** ("positive skew" with stretched right tail and mass concentrated on the left)

PDF of gamma 3.0, 0.5**:**



https://www.teawithtolkien.com/fonts

Example: NORMAL

Parameters: mean (location/position)… real μ,
standard deviation (dispersion/scale)… real $\sigma > 0$

Support: entire real line $(-\infty, \infty)$

Definition: **"ONE DISTRIBUTION FAMILY TO RULE THEM ALL*,**
ONE FAMILY TO APPROXIMATE THEM, ONE FAMILY
TO SUM THEM ALL* AND IN THE LIMIT BE THEM"

PDF at x: $1 / (\sqrt{2\pi}\cdot\sigma\cdot e^{[(x-\mu)/\sigma]^2/2})$, which is symmetric around μ; *R:* `dnorm(…)`
$z := (x-\mu)/\sigma$ same as **STANDARD NORMAL** case $\mu=0, \sigma=1$

PDF at z: $1 / (\sqrt{2\pi}\cdot e^{z^2/2})$,

CDF at z: $(1 + \underline{\text{erf}}(z/\sqrt{2}))/2 =$
$1 - (\underline{\text{erfc}}(z/\sqrt{2})/2)$,
$\approx$95% in $\mu\pm2\sigma$, $\approx$99.7% in $\mu\pm3\sigma$

Mean: μ

Std. deviation: σ

Quartiles: $\approx \mu-0.67\cdot\sigma$, $= \mu$, $\approx \mu+0.67\cdot\sigma$
R: `qnorm(…)`

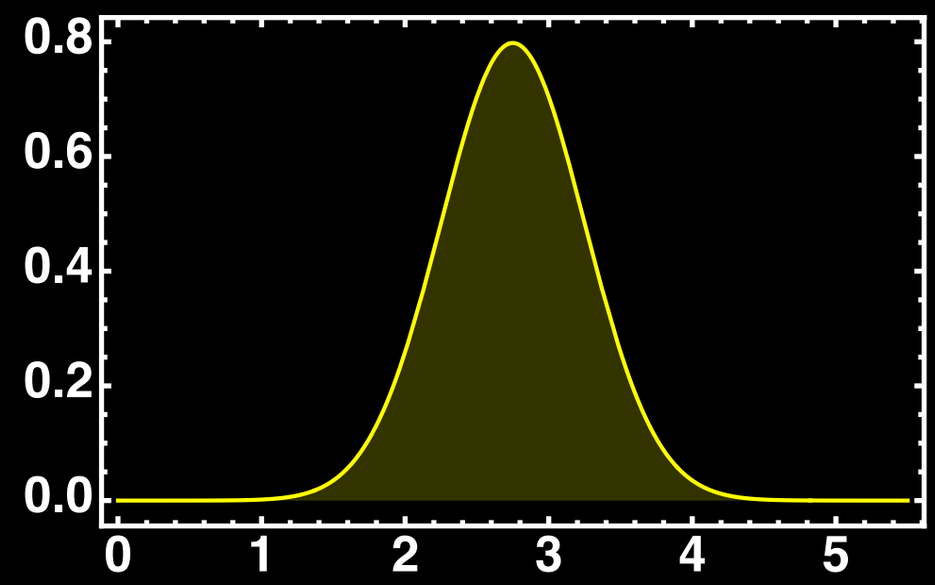
Modes: μ

Sometimes we let $\sigma = 0$ and take as a unit point mass at μ

R: `rnorm(…)` *simulates*

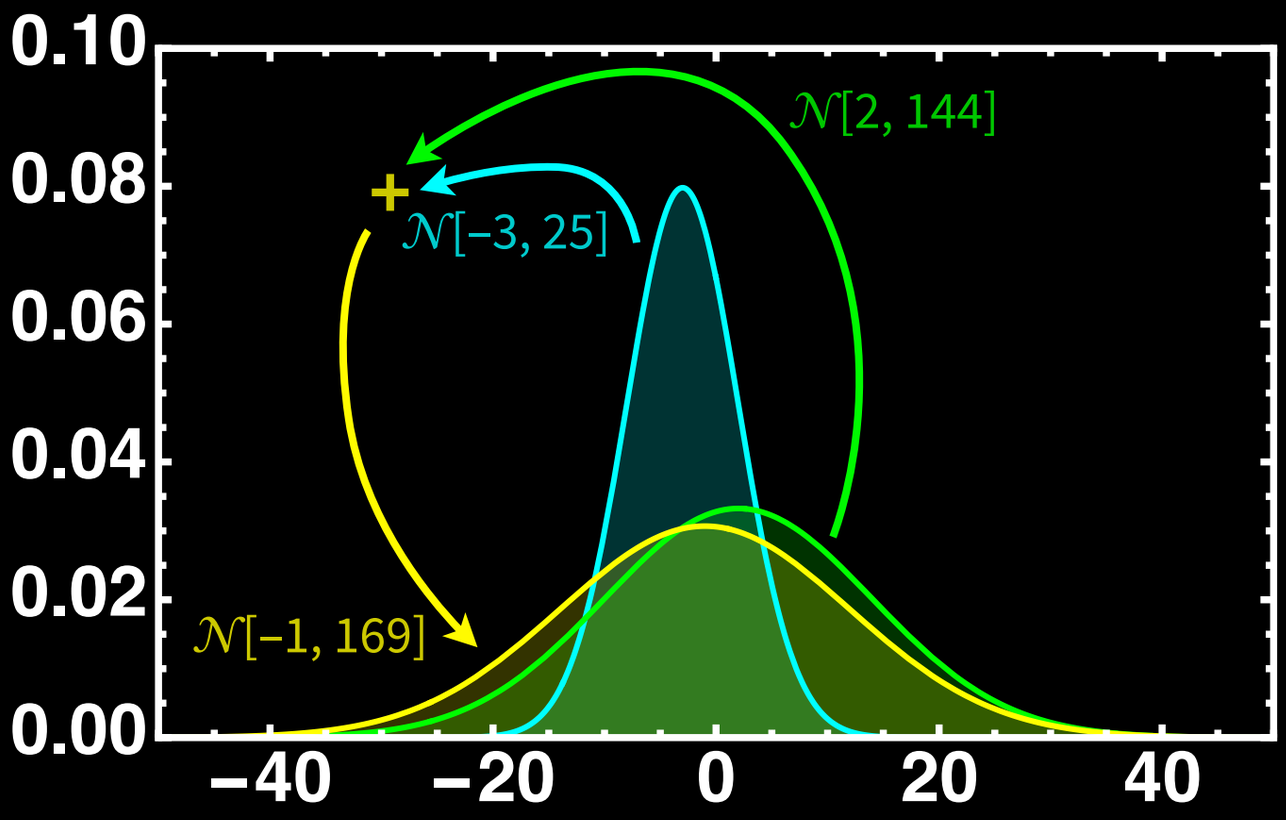
R: `pnorm(…)`

PDF of normal 2.75, 0.5**:**



Add/Scale/Shift/Flip Normal Stays Normal

- Any affine ("linear") combo. of INDEPENDENT normals still normal … **write $\mathcal{N}$[mean, VARIANCE] for a normal:**
  - With c_0, c_1, …, c_m any real constants:
    $c_0 + c_1\,\mathcal{N}[\mu_1, \sigma_1^2] + \ldots + c_m\,\mathcal{N}[\mu_m, \sigma_m^2]$ distributed
    as $\mathcal{N}[c_0 + c_1\,\mu_1 + \ldots + c_m\,\mu_m,\ \ c_1^2\,\sigma_1^2 + \ldots + c_m^2\,\sigma_m^2]$
    NOTE: INDEPENDENT VARIANCES ALWAYS ADD



Std. devs. 12, 5 →
Variance $12^2 + 5^2$ →
Std. dev. $\sqrt{12^2 + 5^2} = 13$

Ubiquitous in Sums

- **CENTRAL LIMIT THEOREM (CLT)…** summing **independent** observations *approaches*** *a normal distribution* NO MATTER WHAT* THE SUMMANDS' DISTRIBUTIONS ARE

Ubiquitous in Sums

- **CENTRAL LIMIT THEOREM (CLT)…** summing **independent** observations *approaches*** *a normal distribution* NO MATTER WHAT* THE SUMMANDS' DISTRIBUTIONS ARE

THERE ARE TWO BIG CAVEATS:

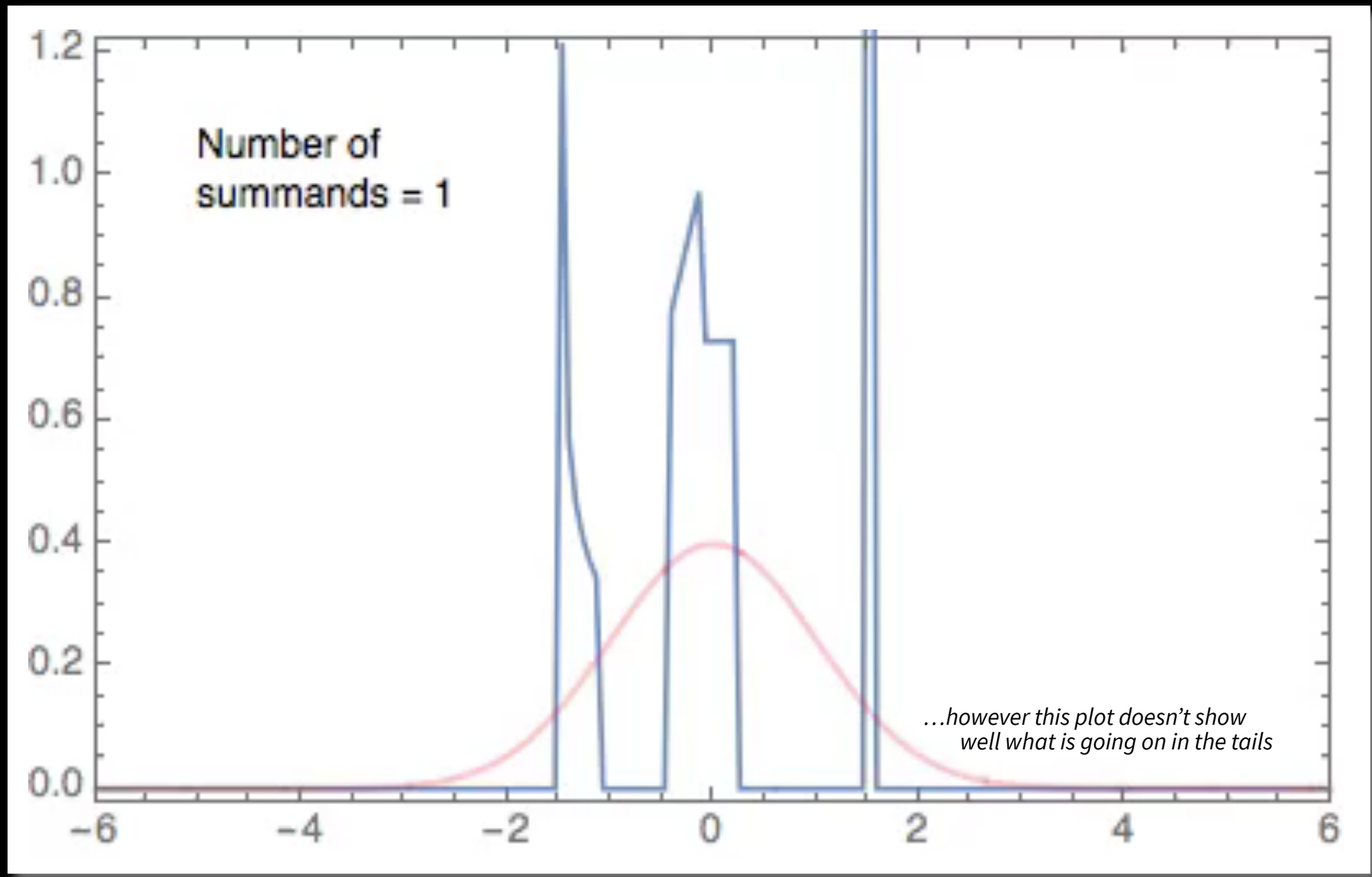
****This is an *asymptotic* statement** that requires the infinite limit to get arbitrarily close in full generality — IN PRACTICE, THE SPEED OF APPROXIMATION IS VERY IMPORTANT IN APPLICATIONS, *and the needed sample sizes to have high probability of having a specific property being approximated to a specific degree depend heavily on the property, desired accuracy, and THE SUMMANDS DISTRIBUTIONS!*

*As long as the summand distributions don't change too much and are not constants and don't have 'too much' probability going out toward infinity… *for example: not every distribution has a variance, or a mean*

*Approx. often gets suff. good for many purposes involving "middles" of distns. relatively fast (say, samp. size 20 or 30) …BUT CONVERGES **VERY SLOW** IN TAILS*

Example of CLT Convergence to Normal

Scaled/shifted sum of 1..20 i.i.d. copies of distribution in first frame

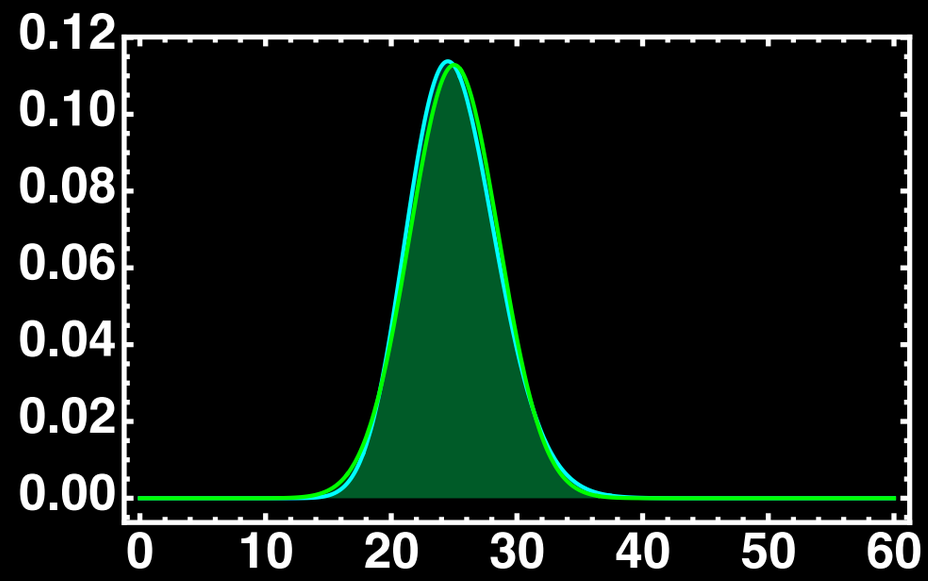


If your data are too non-normal for your specific test/model/situation assumptions, sometimes you can transform it (e.g., log, ranks, …) to make it sufficiently well-behaved

Example: Sums of Exponentials

- **How to tell what normal mean and variance approximates a given sum?**
  - **Means add and variances add** *when adding independent anythings*
  - Suppose have a size 50 random sample $x_1, \ldots, x_{50}$ ~ Exponential[2.0]
    - Exponential[2.0] has mean 0.5, variance 0.25
    - Sum 50? Multiply by 50: mean 25, variance 12.5
      THUS… $s := x_1 + \ldots + x_{50} \approx \mathcal{N}[25, 12.5] =: n$
  - Compare: we actually know $s \sim_{\text{exactly}} \text{Gamma}[50, 1/2.0]$

PDF: gamma *s* vs. normal *n*

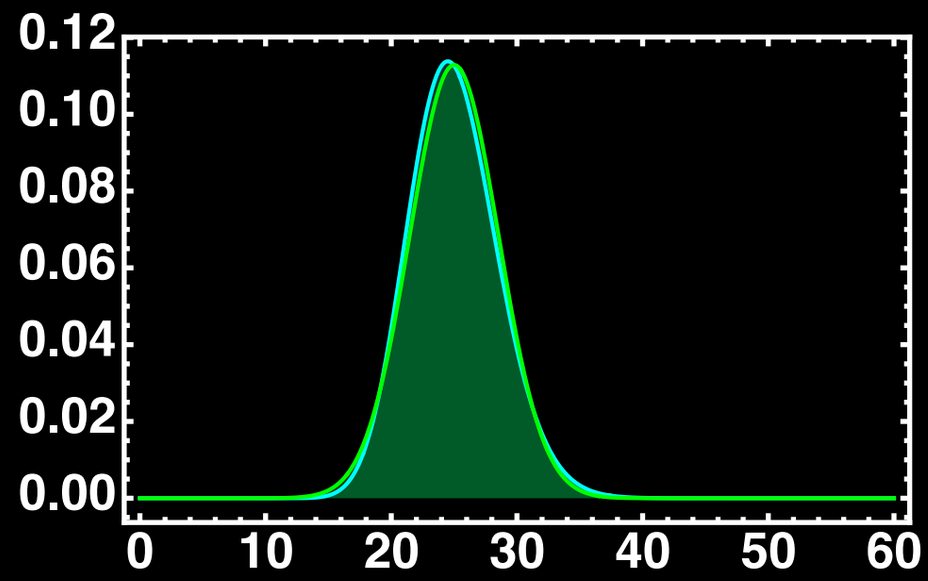


Example: Sums of Exponentials

- **How to tell what normal mean and variance approximates a given sum?**
  - **Means add and variances add** *when adding independent anythings*
  - Suppose have a size 50 random sample $x_1, \ldots, x_{50}$ ~ Exponential[2.0]
    - Exponential[2.0] has mean 0.5, variance 0.25
    - Sum 50? Multiply by 50: mean 25, variance 12.5
      THUS… $s := x_1 + \ldots + x_{50} \approx \mathcal{N}[25, 12.5] =: n$
    - Compare: we actually know $s \sim_{\text{exactly}} \text{Gamma}[50, 1/2.0]$

- **MAYBE YOU SEE WHERE THIS CAN GO?**
  …SAMPLE MEAN IS JUST A RESCALED SUM!

PDF: gamma *s* vs. normal *n*



Example: Heights of Humans

- **Height** affected by many factors
  - Lots of genetic loci with many alleles, lots of environmental factors; *presumably many acting independently*

    Why many macroscopic phenotypes in biology approximately normal
  - R package `gamlss.data` has dataset `dbhh` giving head circumference (cm), age (years: 0.03 to 21.68), and height (cm) for 6,885 Dutch boys from the Fourth Dutch Growth Study
    - SJC exported `gamlss.data::dbhh` to a GZIP'd tab-sep. plain text file at URL https://zfishskin.net/Private/RsampleData--gamlss.data--dbhh--forQCbioW14.tsv.gz

caseId	head	age	ht
1	33.6	0.03	53.3
2	33.6	0.04	50.8
……	……	……	……
7481	58.5	21.68	186.5

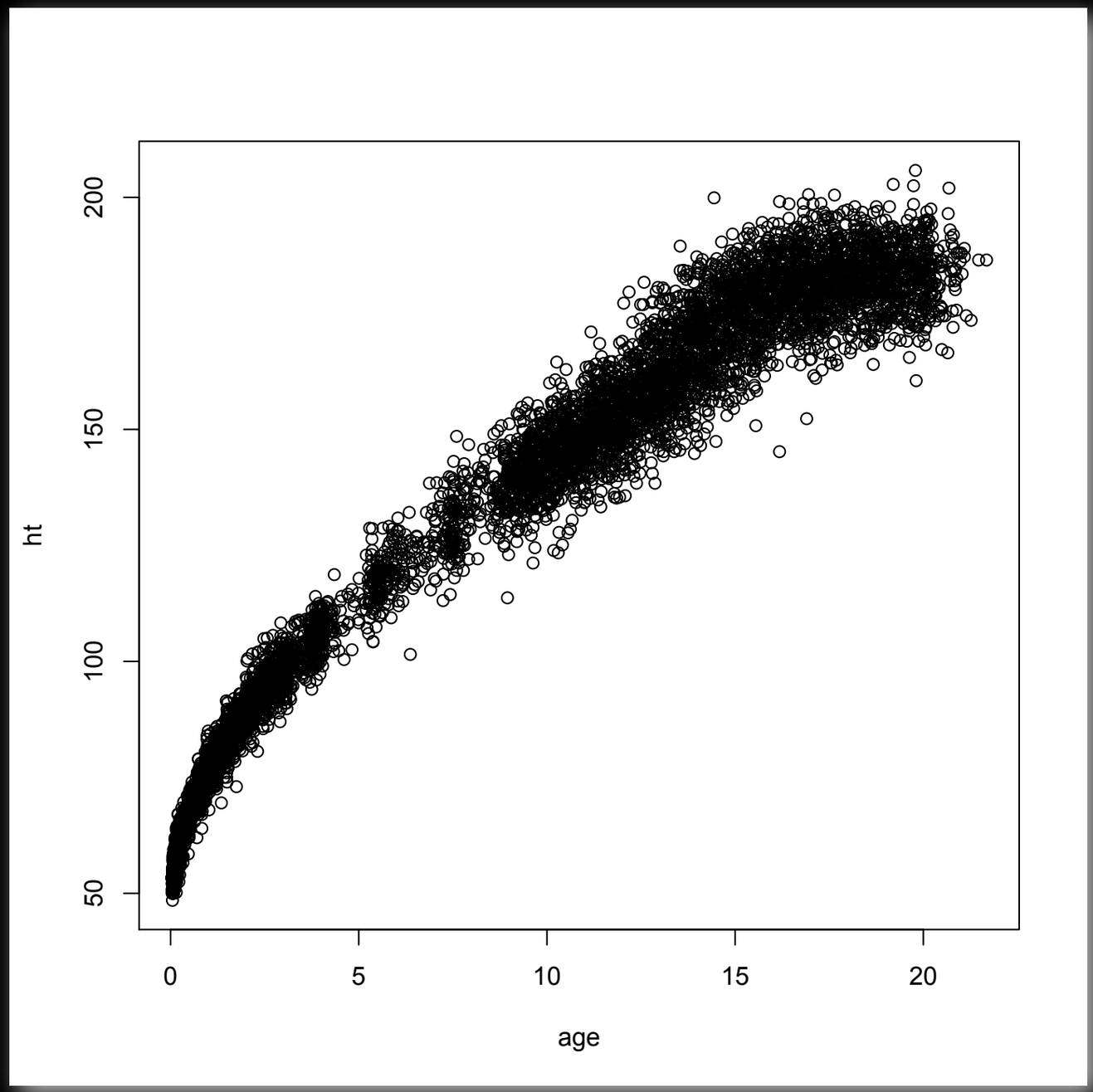
```
dBoys <- read.table(
           gzcon(url(paste0(
                       "https://zfishskin.net/Private/RsampleData--",
                       "gamlss.data--dbhh--forQCbioW14.tsv.gz")       ),
             text=TRUE),
           header=TRUE, sep="\t", quote="", row.names="caseId",
           colClasses=c(caseId='integer', head='numeric',
                          age='numeric',   ht='numeric'  ),
           comment.char="", allowEscapes=FALSE);
```

Example: Heights of Humans

```
str(dBoys)
'data.frame': 6885 obs. of  3 variables:
 $ head: num  33.6 33.6 33.7 35 36.1 36.6 38 37.4 35.1 36.8 ...
 $ age : num  0.03 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 ...
 $ ht  : num  53.3 50.8 50.1 53.5 52.1 50.5 52.5 55 52.9 50 ...
summary(dBoys)
      head            age              ht
 Min.   :33.50   Min.   : 0.030   Min.   : 48.5
 1st Qu.:48.90   1st Qu.: 1.810   1st Qu.: 86.5
 Median :53.00   Median :10.170   Median :143.4
 Mean   :51.73   Mean   : 9.099   Mean   :131.1
 3rd Qu.:55.80   3rd Qu.:14.960   3rd Qu.:172.7
 Max.   :66.30   Max.   :21.680   Max.   :205.8
```

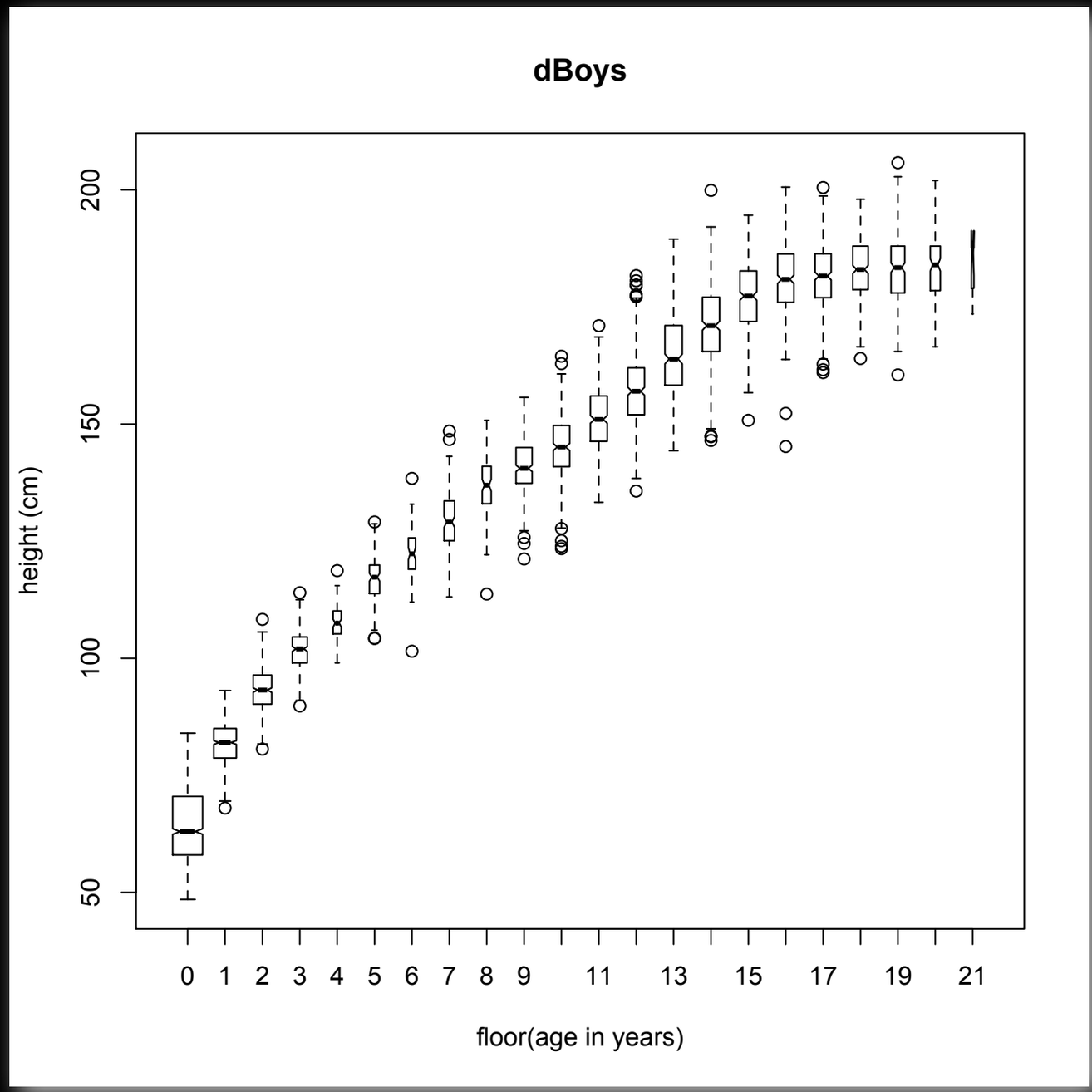
Example: Heights of Humans

```
str(dBoys)
'data.frame': 6885 obs. of  3 variables:
 $ head: num  33.6 33.6 33.7 35 36.1 36.6 38 37.4 35.1 36.8 ...
 $ age : num  0.03 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 ...
 $ ht  : num  53.3 50.8 50.1 53.5 52.1 50.5 52.5 55 52.9 50 ...
summary(dBoys)
…       age                ht
… Min.   : 0.030   Min.   : 48.5
… 1st Qu.: 1.810   1st Qu.: 86.5
… Median :10.170   Median :143.4
… Mean   : 9.099   Mean   :131.1
… 3rd Qu.:14.960   3rd Qu.:172.7
… Max.   :21.680   Max.   :205.8
plot(ht ~ age, data=dBoys);
```



Example: Heights of Humans

```
str(dBoys)
'data.frame': 6885 obs. of  3 variables:
 $ head: num  33.6 33.6 33.7 35 36.1 36.6 38 37.4 35.1 36.8 ...
 $ age : num  0.03 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 ...
 $ ht  : num  53.3 50.8 50.1 53.5 52.1 50.5 52.5 55 52.9 50 ...
summary(dBoys)
…       age              ht
… Min.   : 0.030   Min.   : 48.5
… 1st Qu.: 1.810   1st Qu.: 86.5
… Median :10.170   Median :143.4
… Mean   : 9.099   Mean   :131.1
… 3rd Qu.:14.960   3rd Qu.:172.7
… Max.   :21.680   Max.   :205.8
plot(ht ~ age, data=dBoys);
boxplot(ht ~ floor(age), data=dBoys,
  varwidth=TRUE, notch=TRUE,
  xlab='floor(age in years)',
  ylab='height (cm)', main='dBoys')
```



Example: Heights of Humans

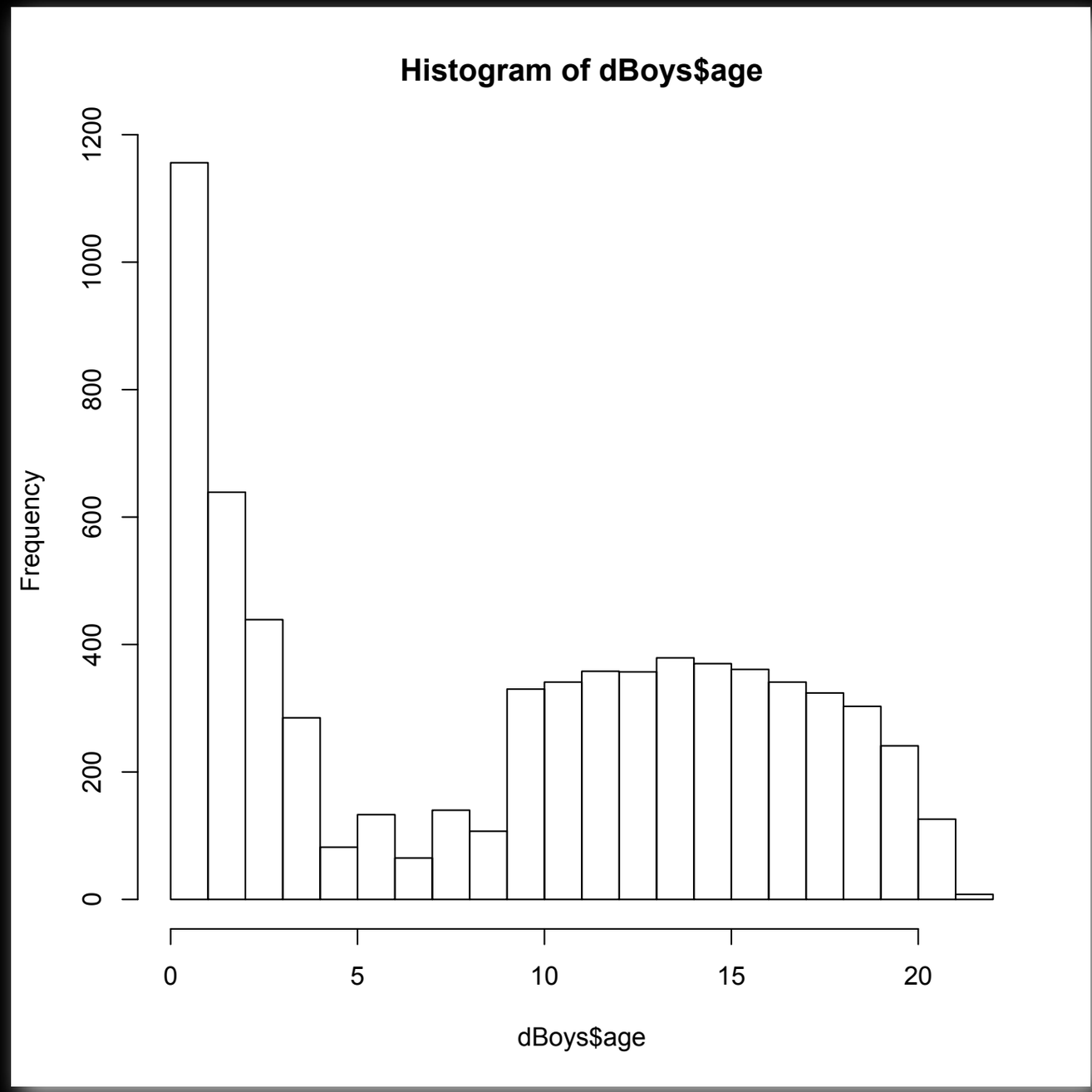
```
str(dBoys)
'data.frame': 6885 obs. of  3 variables:
 $ head: num  33.6 33.6 33.7 35 36.1 36.6 38 37.4 35.1 36.8 ...
 $ age : num  0.03 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 ...
 $ ht  : num  53.3 50.8 50.1 53.5 52.1 50.5 52.5 55 52.9 50 ...
```

```
summary(dBoys)
…       age                ht
… Min.   : 0.030   Min.   : 48.5
… 1st Qu.: 1.810   1st Qu.: 86.5
… Median :10.170   Median :143.4
… Mean   : 9.099   Mean   :131.1
… 3rd Qu.:14.960   3rd Qu.:172.7
… Max.   :21.680   Max.   :205.8
```

```
plot(ht ~ age, data=dBoys);
```

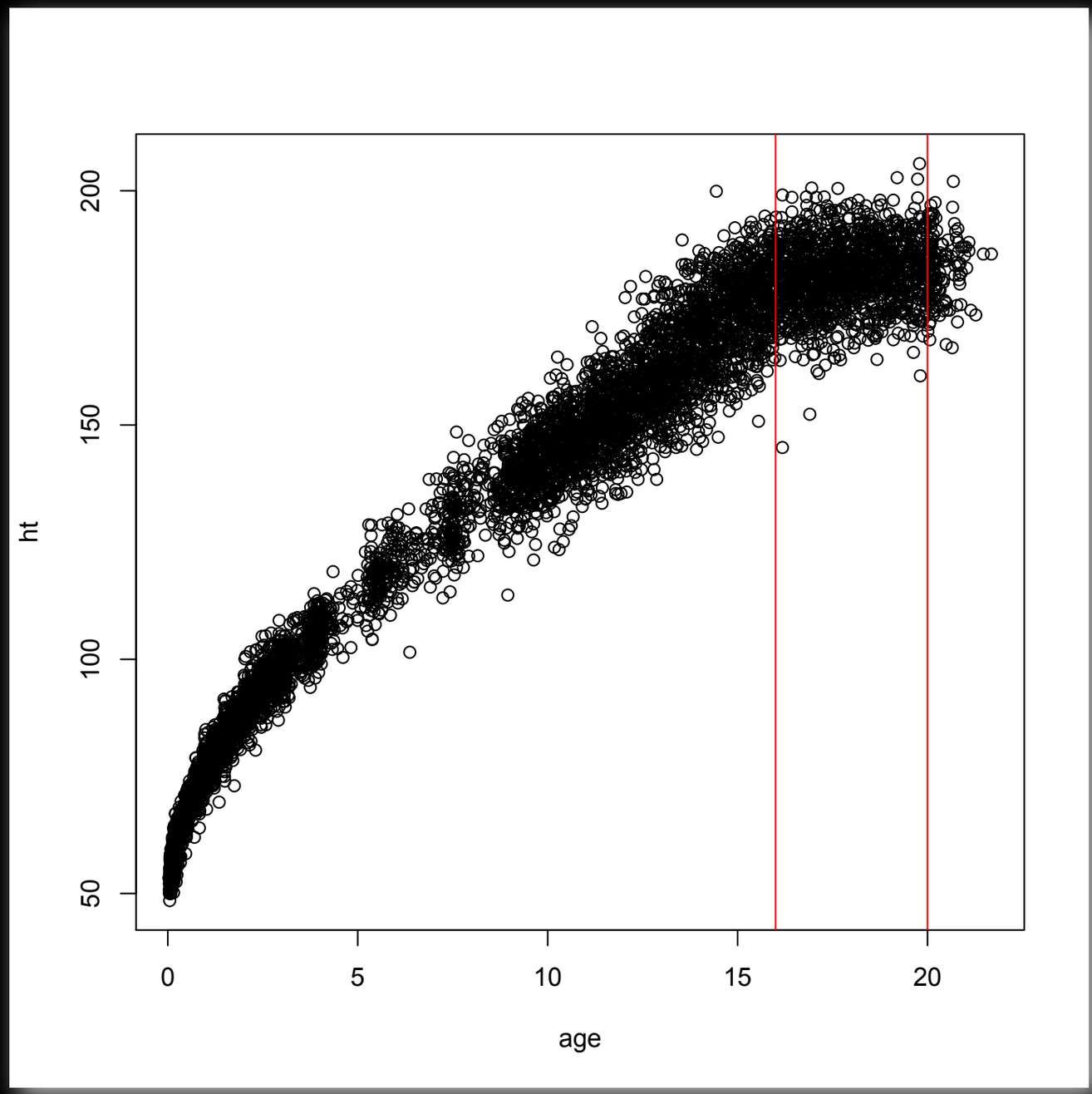
```
boxplot(ht ~ floor(age), data=dBoys,
  varwidth=TRUE, notch=TRUE,
  xlab='floor(age in years)',
  ylab='height (cm)', main='dBoys')
```

```
hist(dBoys$age, breaks=0:22);
```



Example: Heights of Humans

```
str(dBoys)
'data.frame': 6885 obs. of  3 variables:
 $ head: num  33.6 33.6 33.7 35 36.1 36.6 38 37.4 35.1 36.8 ...
 $ age : num  0.03 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 ...
 $ ht  : num  53.3 50.8 50.1 53.5 52.1 50.5 52.5 55 52.9 50 ...
summary(dBoys)
…       age                ht
… Min.   : 0.030   Min.   : 48.5
… 1st Qu.: 1.810   1st Qu.: 86.5
… Median :10.170   Median :143.4
… Mean   : 9.099   Mean   :131.1
… 3rd Qu.:14.960   3rd Qu.:172.7
… Max.   :21.680   Max.   :205.8
plot(ht ~ age, data=dBoys);
boxplot(ht ~ floor(age), data=dBoys,
  varwidth=TRUE, notch=TRUE,
  xlab='floor(age in years)',
  ylab='height (cm)', main='dBoys')
hist(dBoys$age, breaks=0:22);
plot(ht ~ age, data=dBoys);
  abline(v=16.0, col="red");
  abline(v=20.0, col="red");
```



Example: Heights of Humans

```
str(dBoys)
```

```
'data.frame': 6885 obs. of  3 variables:
 $ head: num  33.6 33.6 33.7 35 36.1 36.6 38 37.4 35.1 36.8 ...
 $ age : num  0.03 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 ...
 $ ht  : num  53.3 50.8 50.1 53.5 52.1 50.5 52.5 55 52.9 50 ...
```

```
summary(dBoys)
```

```
      head            age              ht
 Min.   :33.50   Min.   : 0.030   Min.   : 48.5
 1st Qu.:48.90   1st Qu.: 1.810   1st Qu.: 86.5
 Median :53.00   Median :10.170   Median :143.4
 Mean   :51.73   Mean   : 9.099   Mean   :131.1
 3rd Qu.:55.80   3rd Qu.:14.960   3rd Qu.:172.7
 Max.   :66.30   Max.   :21.680   Max.   :205.8
```

```
plot(ht ~ age, data=dBoys);
```

```
boxplot(ht ~ floor(age), data=dBoys,
  varwidth=TRUE, notch=TRUE,
  xlab='floor(age in years)',
  ylab='height (cm)', main='dBoys')
```

```
hist(dBoys$age, breaks=0:22);
```

```
plot(ht ~ age, data=dBoys);
  abline(v=16.0, col="red");
  abline(v=20.0, col="red");
```

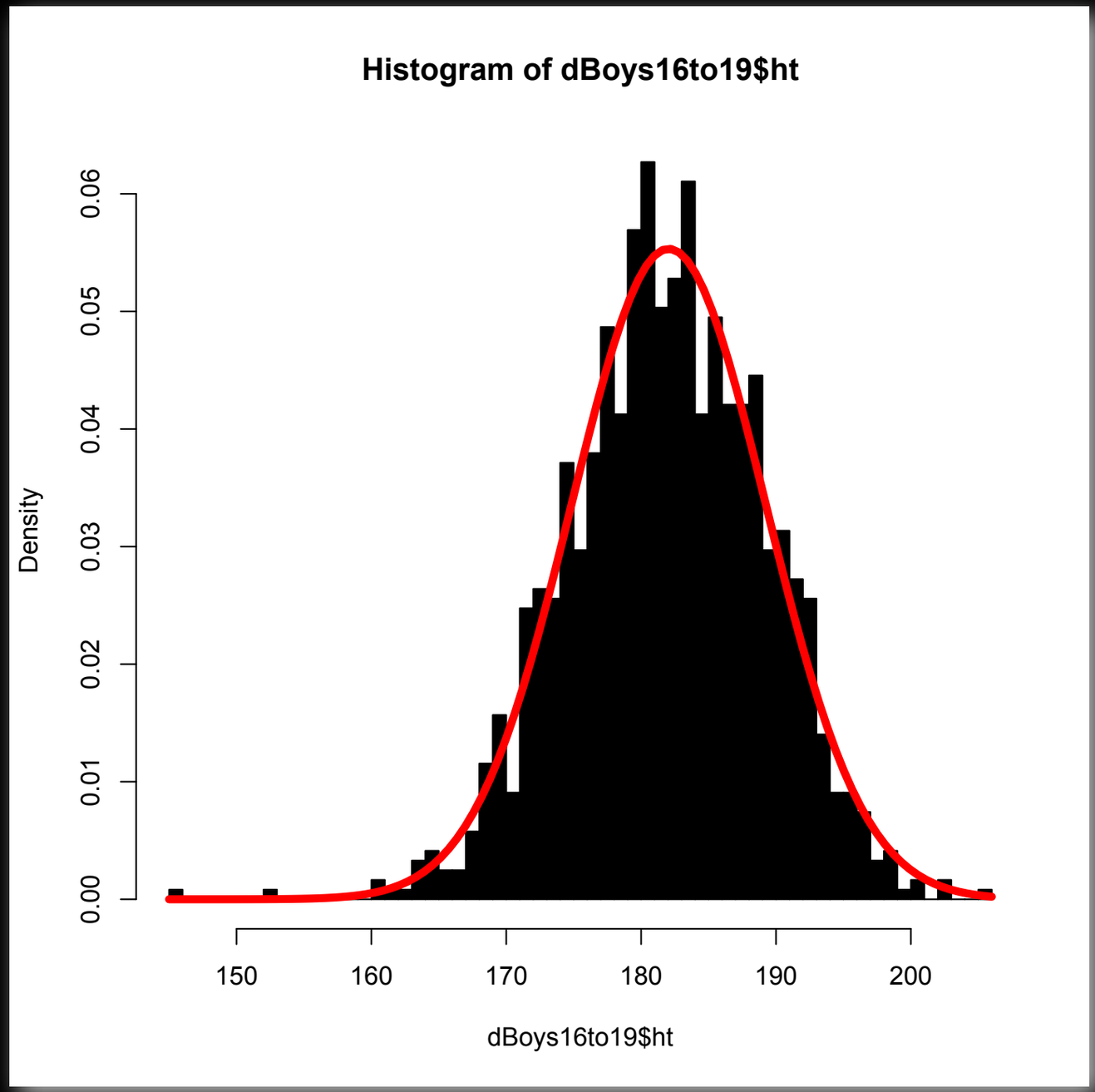
```
dBoys16to19 <-
  dBoys[dBoys$age >= 16.0  &
        dBoys$age <  20.0,   ];
nrow(dBoys16to19)  # 1212
```

```
summary(dBoys16to19)
```

```
      head            age              ht
 Min.   :50.00   Min.   :16.00   Min.   :145.2
 1st Qu.:56.00   1st Qu.:16.92   1st Qu.:177.3
 Median :57.00   Median :17.77   Median :182.0
 Mean   :57.04   Mean   :17.86   Mean   :182.0
 3rd Qu.:58.20   3rd Qu.:18.76   3rd Qu.:187.1
 Max.   :66.30   Max.   :19.99   Max.   :205.8
```

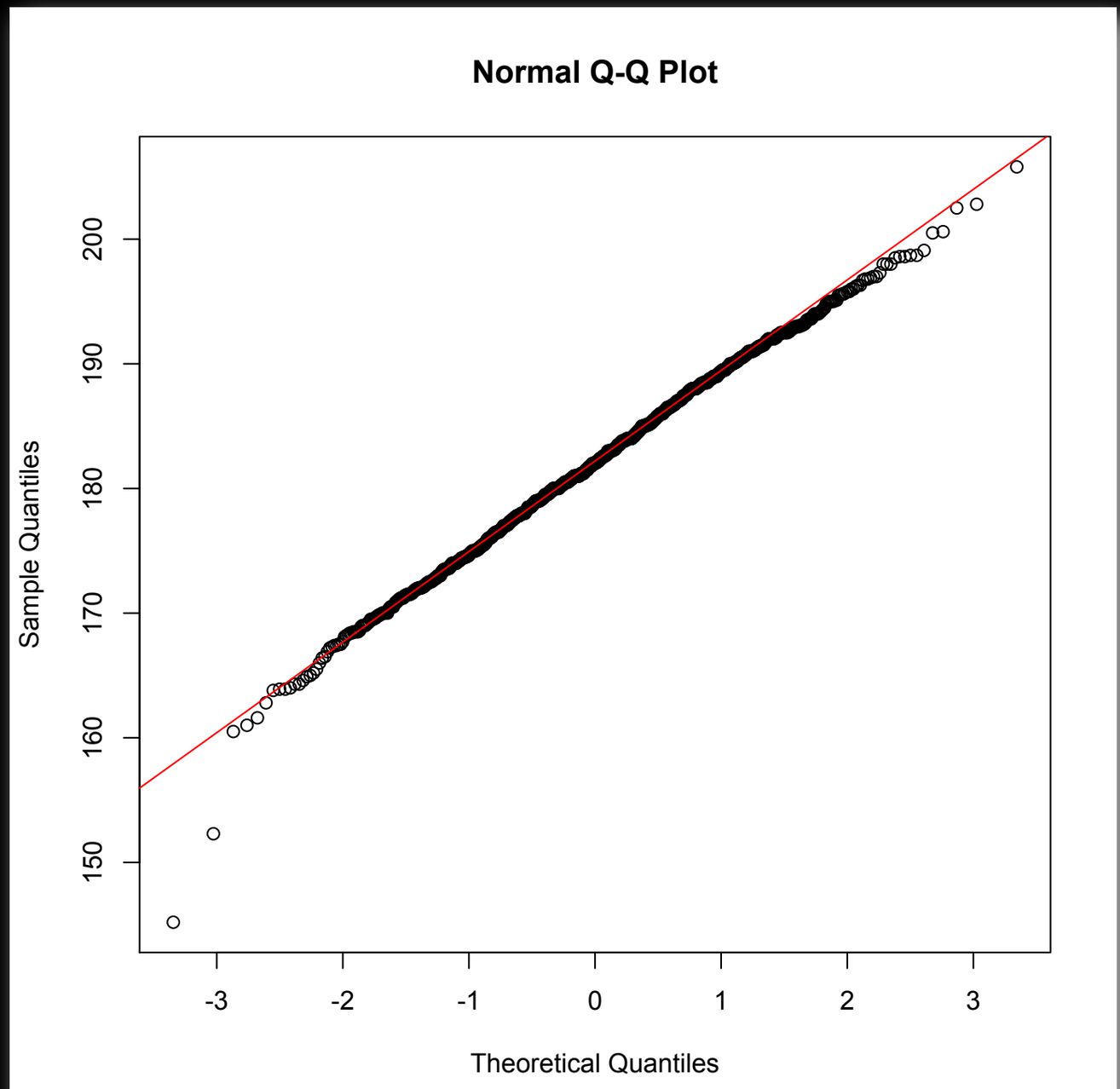
Example: Heights of Humans

```
(dMu <- mean(dBoys16to19$ht))  # 182.0427
(dSD <-   sd(dBoys16to19$ht))  # 7.209898
hist(dBoys16to19$ht, breaks=145:206, freq=FALSE, col="black");
  curve(dnorm(x, mean=dMu, sd=dSD),
    from=145, to=206, col="red", lwd=5, add=TRUE);
```



Example: Heights of Humans

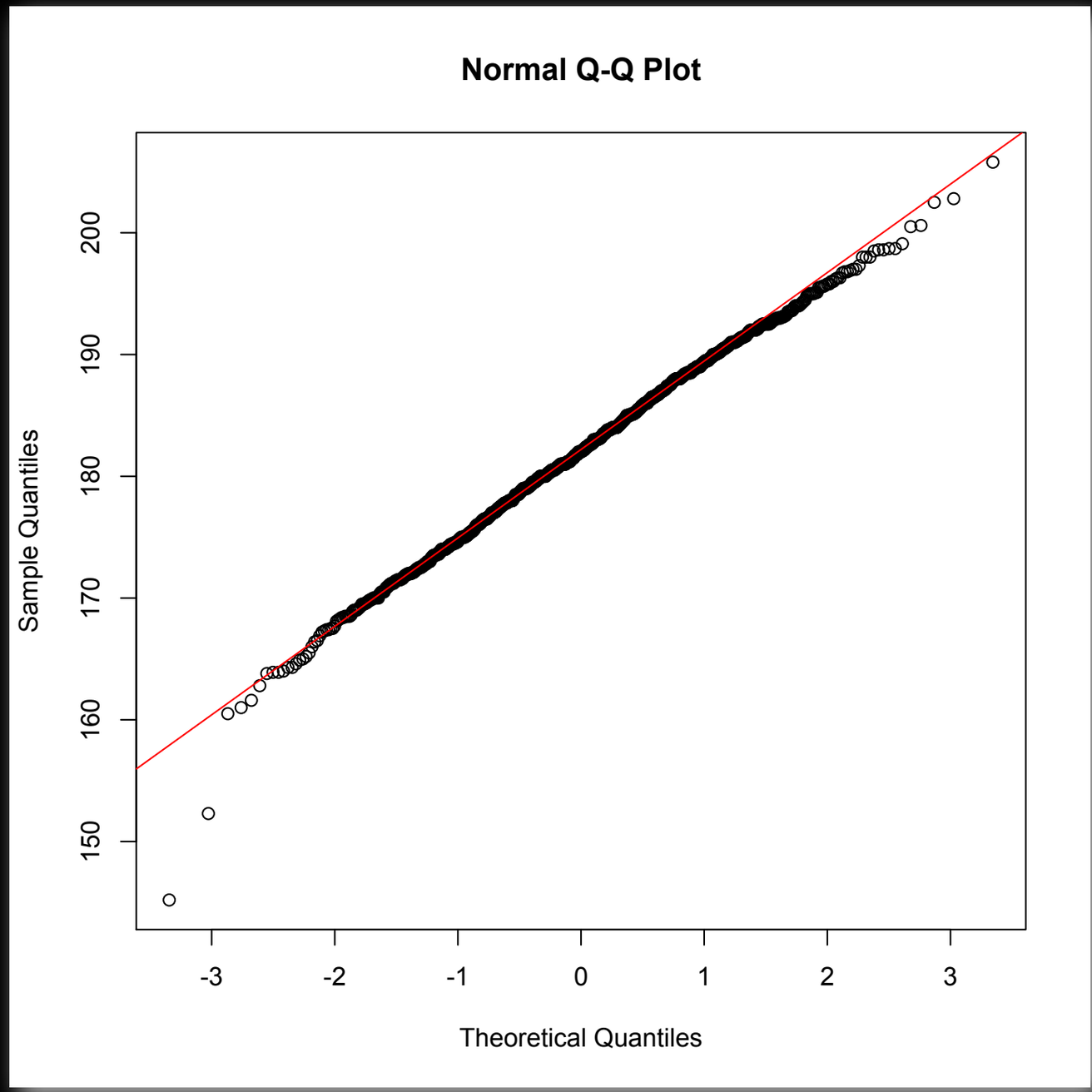
```
(dMu <- mean(dBoys16to19$ht))  # 182.0427
(dSD <-   sd(dBoys16to19$ht))  # 7.209898
hist(dBoys16to19$ht, breaks=145:206, freq=FALSE, col="black");
  curve(dnorm(x, mean=dMu, sd=dSD),
    from=145, to=206, col="red", lwd=5, add=TRUE);
qqnorm(dBoys16to19$ht);
  qqline(dBoys16to19$ht,
         col="red");
```



Example: Heights of Humans

```
(dMu <- mean(dBoys16to19$ht))  # 182.0427
(dSD <-   sd(dBoys16to19$ht))  # 7.209898
hist(dBoys16to19$ht, breaks=145:206, freq=FALSE, col="black");
  curve(dnorm(x, mean=dMu, sd=dSD),
    from=145, to=206, col="red", lwd=5, add=TRUE);
qqnorm(dBoys16to19$ht);
  qqline(dBoys16to19$ht,
         col="red");
```

```
shapiro.test(dBoys16to19$ht)
    Shapiro-Wilk normality test
data:  dBoys16to19$ht
W = 0.99558, p-value = 0.001365
```



Example: Heights of Humans

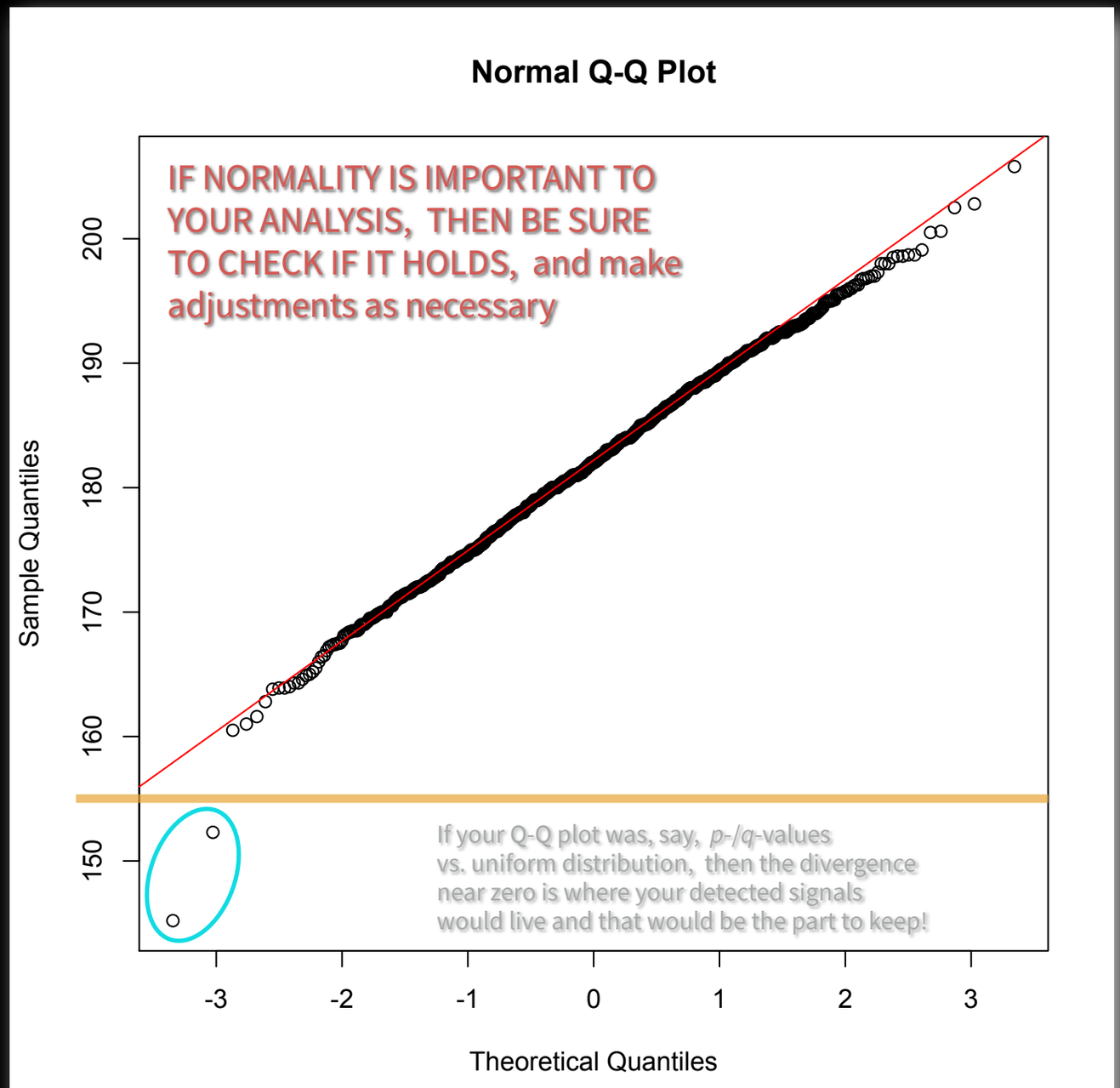
```
(dMu <- mean(dBoys16to19$ht))  # 182.0427
(dSD <-   sd(dBoys16to19$ht))  # 7.209898
hist(dBoys16to19$ht, breaks=145:206, freq=FALSE, col="black");
  curve(dnorm(x, mean=dMu, sd=dSD),
    from=145, to=206, col="red", lwd=5, add=TRUE);
qqnorm(dBoys16to19$ht);
  qqline(dBoys16to19$ht,
         col="red");
```

```
shapiro.test(dBoys16to19$ht)

    Shapiro-Wilk normality test
data:  dBoys16to19$ht
W = 0.99558, p-value = 0.001365
```

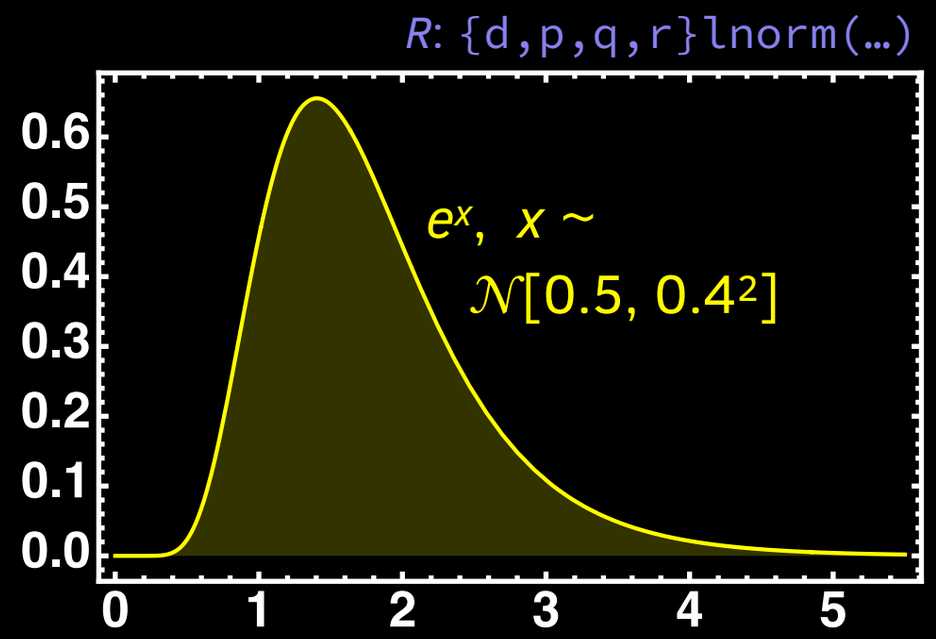
```
# …the two unusually short boys
# (outliers) are driving the
# p-value and normality reject:
shapiro.test(
  dBoys16to19$ht[
    dBoys16to19$ht > 155]);
```

……

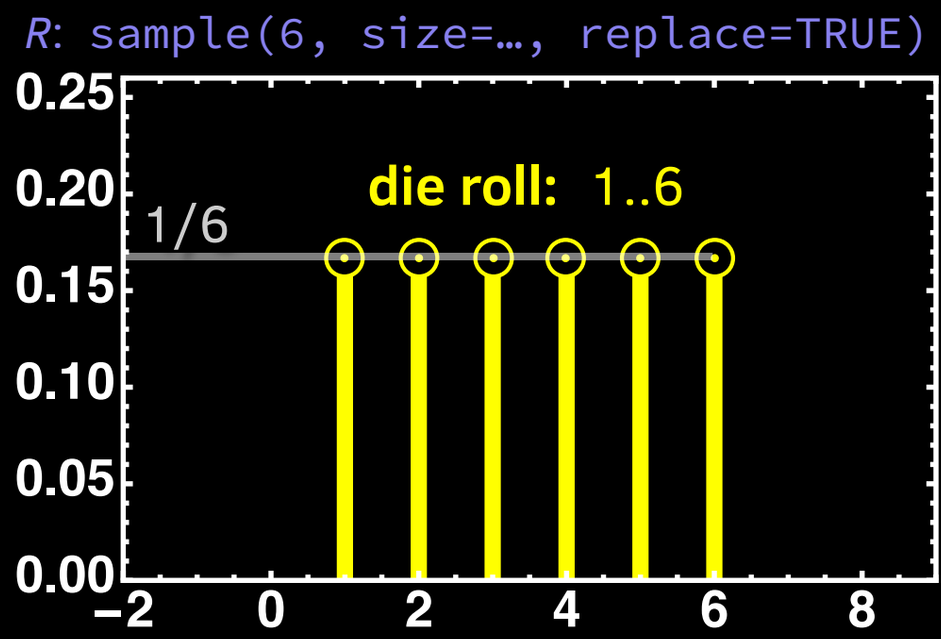
```
W = 0.99878, p-value = 0.5904
```



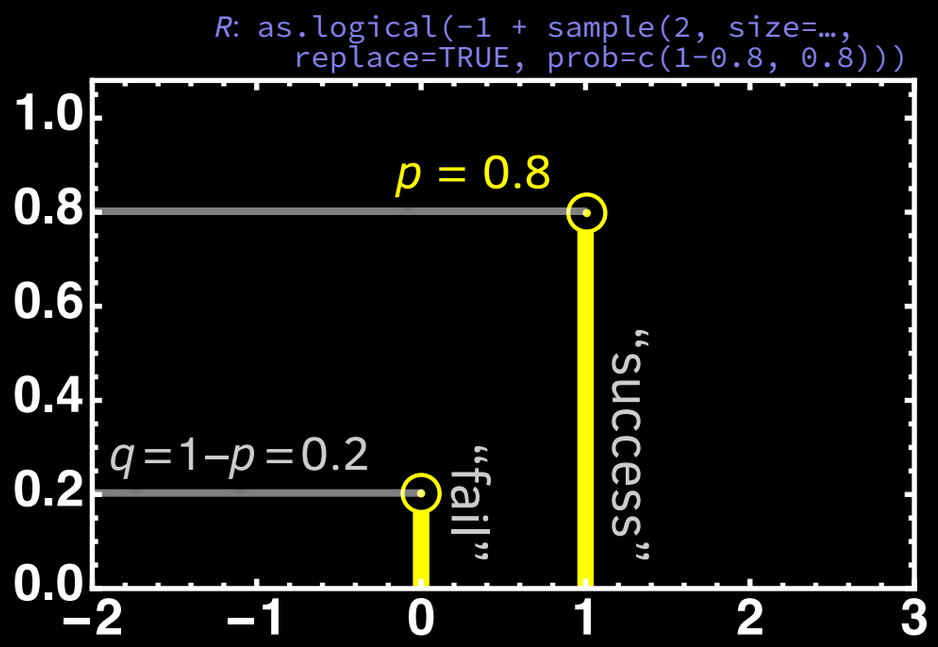
A Few More Example Distributions



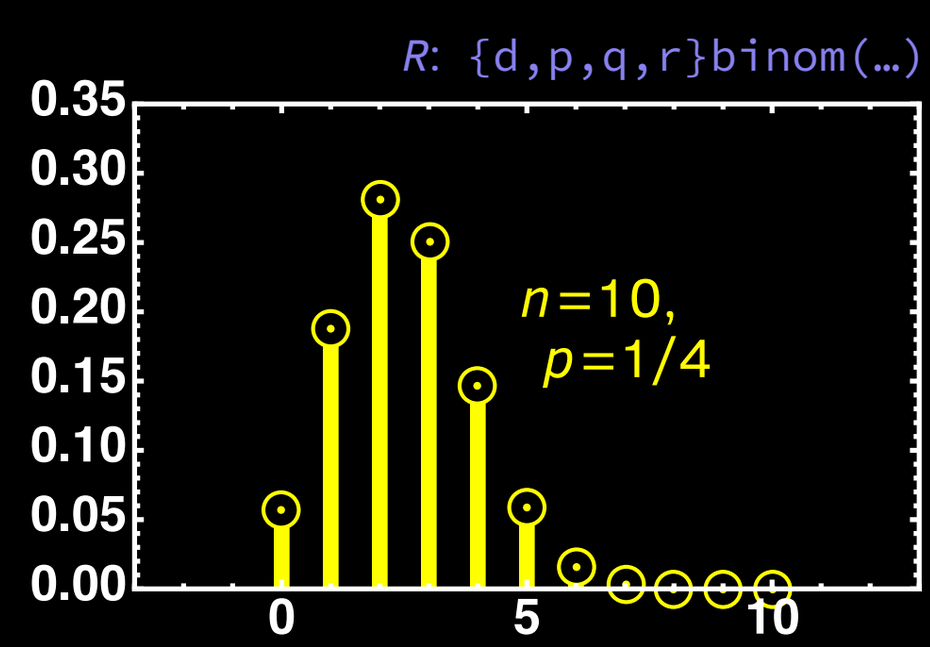
Lognormal: log(observation) ~ a normal; CLT: what *multiplying* independent positive random variables tend toward



Discrete uniform on finite set of items



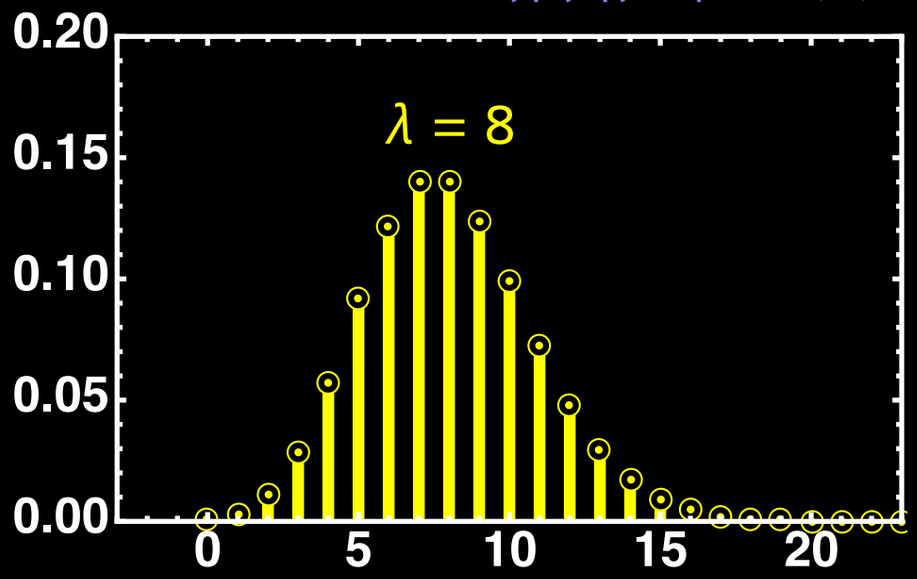
Bernoulli with success probability p



Binomial: # successes in n indep. Bernoulli[p]

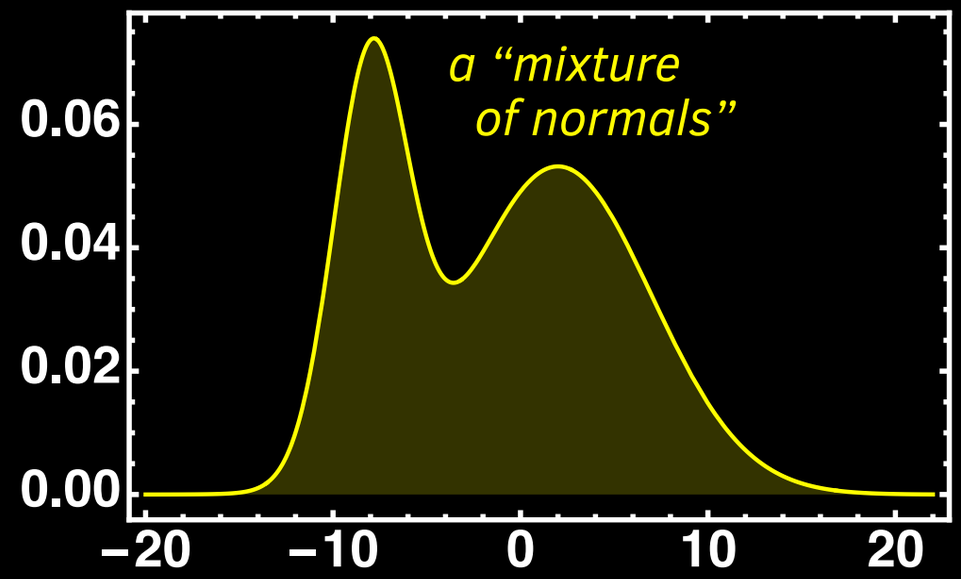
A Few More Example Distributions

R: {d,p,q,r}pois(…)



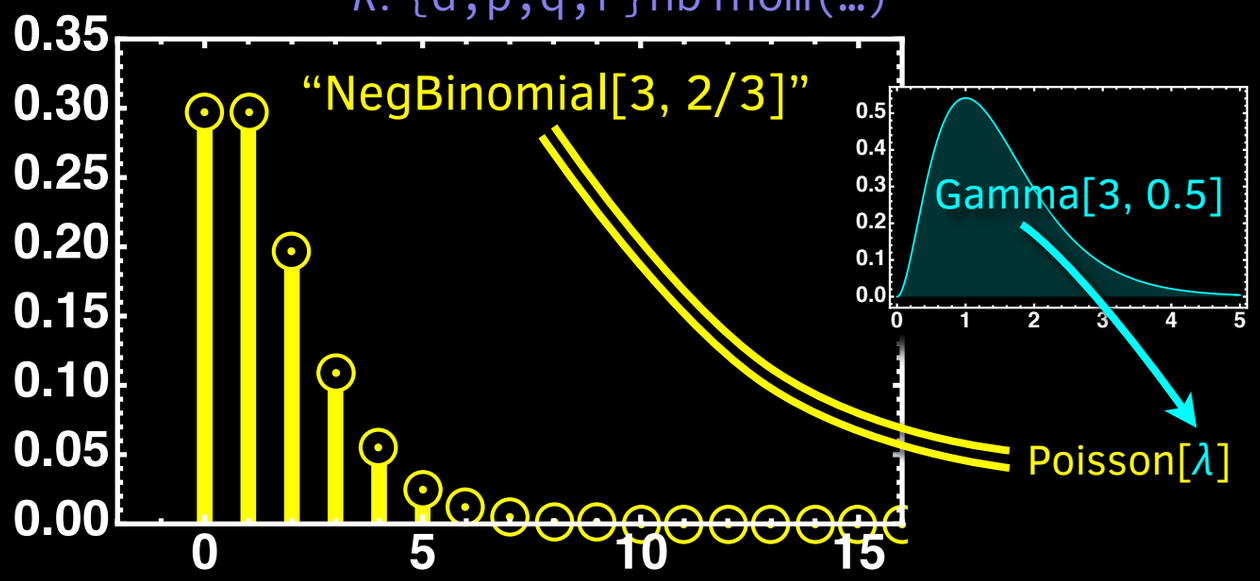
Poisson: # of events per unit time when events uniform at rate λ per unit of time *(compare to gamma)*

```
d <- distr::UnivarMixingDistribution(
       Norm(mean=(-8), sd=2), Norm(mean=2, sd=5),
       mixCoeff=c(1/3.0, 2/3.0));
{d,p,q,r}(d)(…)
```



Finite mixture: example… 1/3 chance from $\mathcal{N}[-8, 2^2]$ and 2/3 chance from $\mathcal{N}[2, 5^2]$

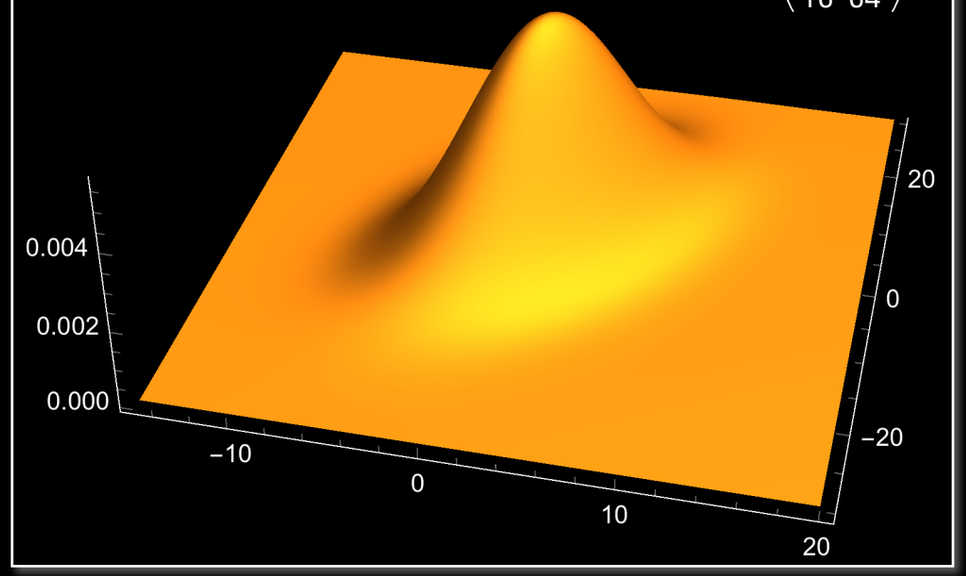
R: {d,p,q,r}nbinom(…)



Compound/continuous mixture: ex. Poisson[λ] when $\lambda \sim$ Gamma[3, 0.5]

```
MASS::mvrnorm(…, c(2,3), matrix(c(16,16, 16,64), 2,2, byrow=TRUE))
```

Binormal[mean (2, 3), std. dev. (4, 8), correlation 0.5] = Multinormal with mean (2, 3) and covariance matrix $\begin{pmatrix} 16 & 16 \\ 16 & 64 \end{pmatrix}$



END OF PART ONE