O'REILLY®

Security

BUILD BETTER DEFENSES

oreillysecuritycon.com

#oreillysecurity

Foundations of Security Data Science

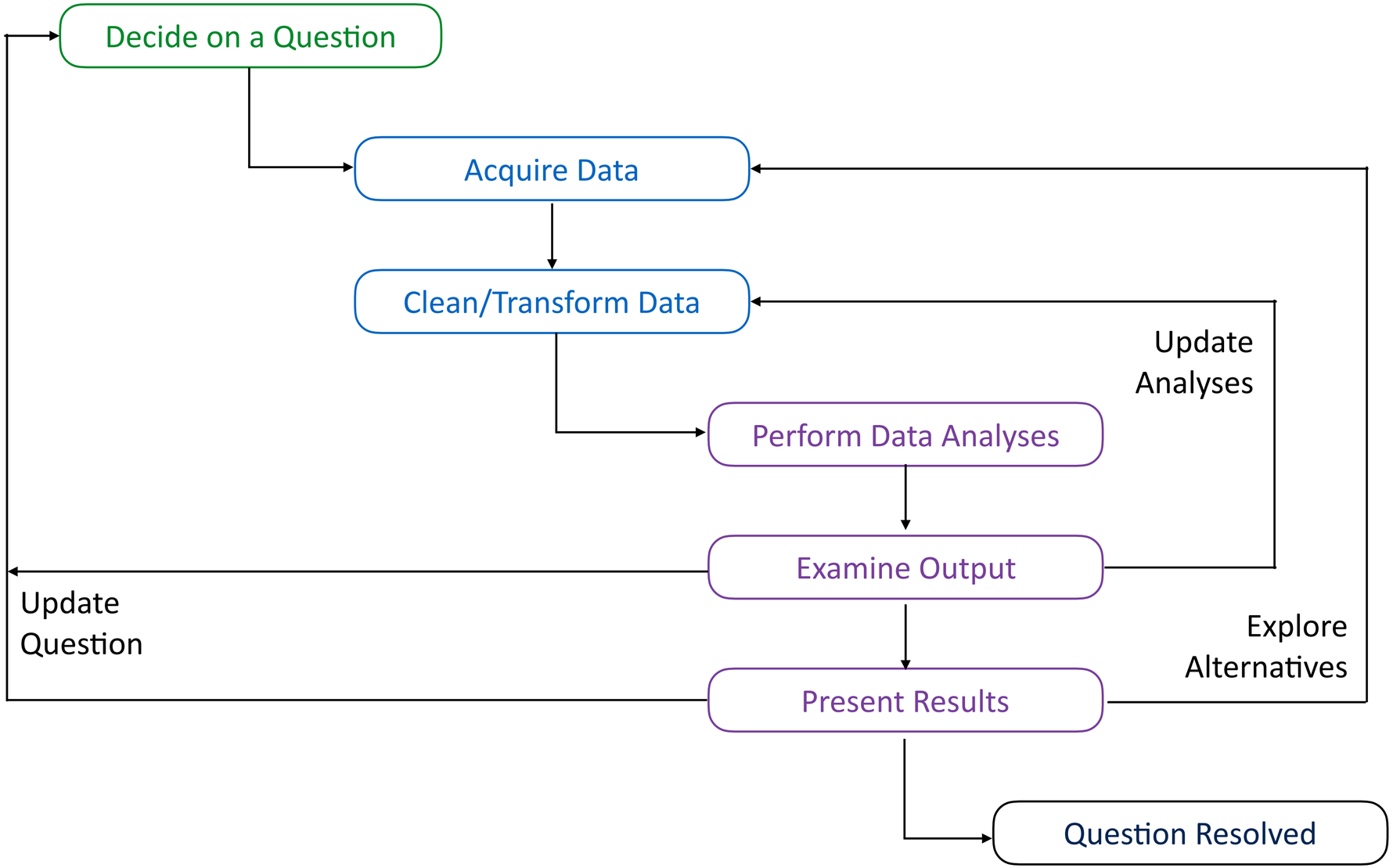
Exploratory Data Analysis

Jay Jacobs
Charles Givre
Bob Rudis

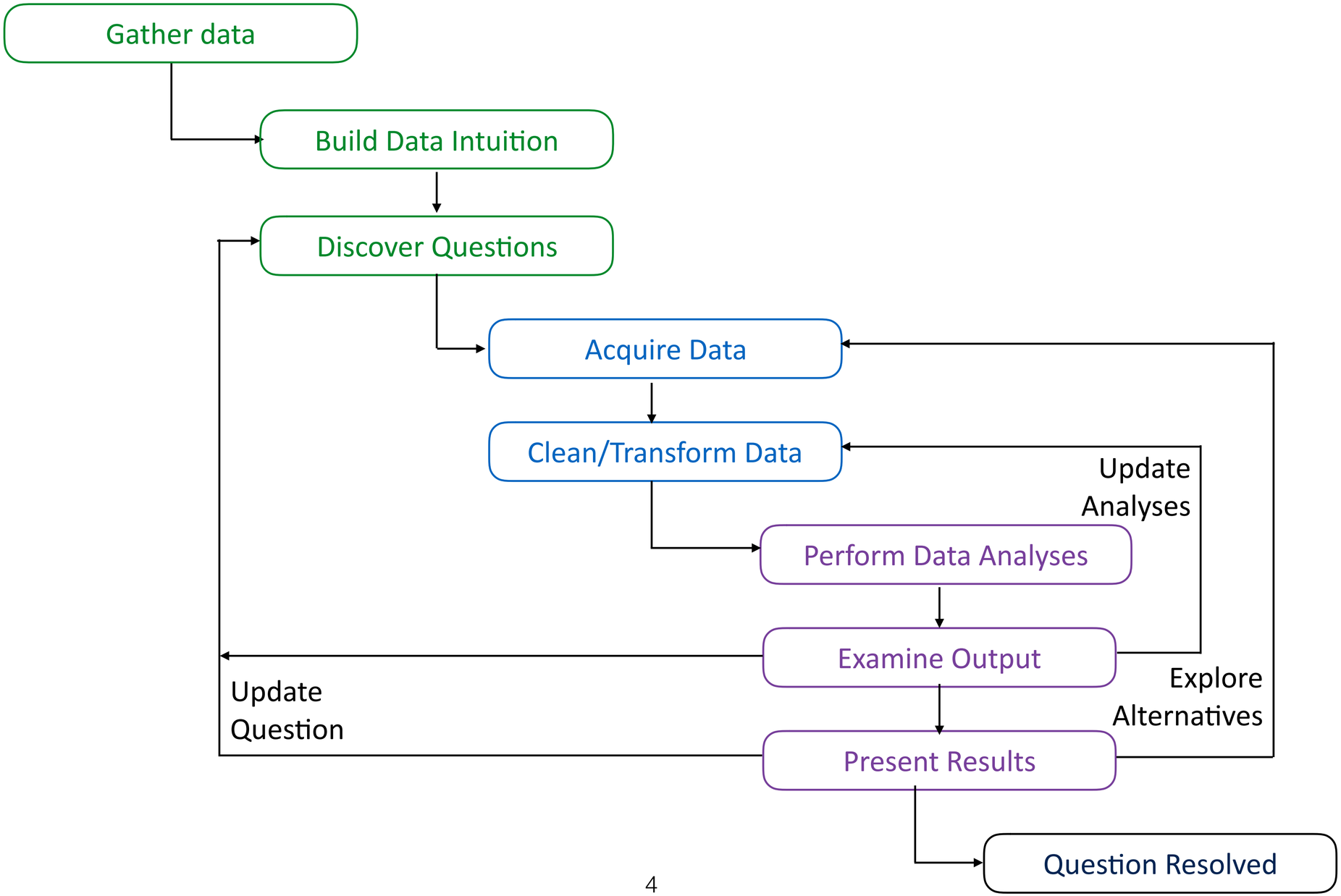
EDA

- “Columnar Thinking”
- Exploratory Data Analysis (EDA) and building your data intuition
- Visualization techniques to describe and demystify the data
- EDA with moving pictures

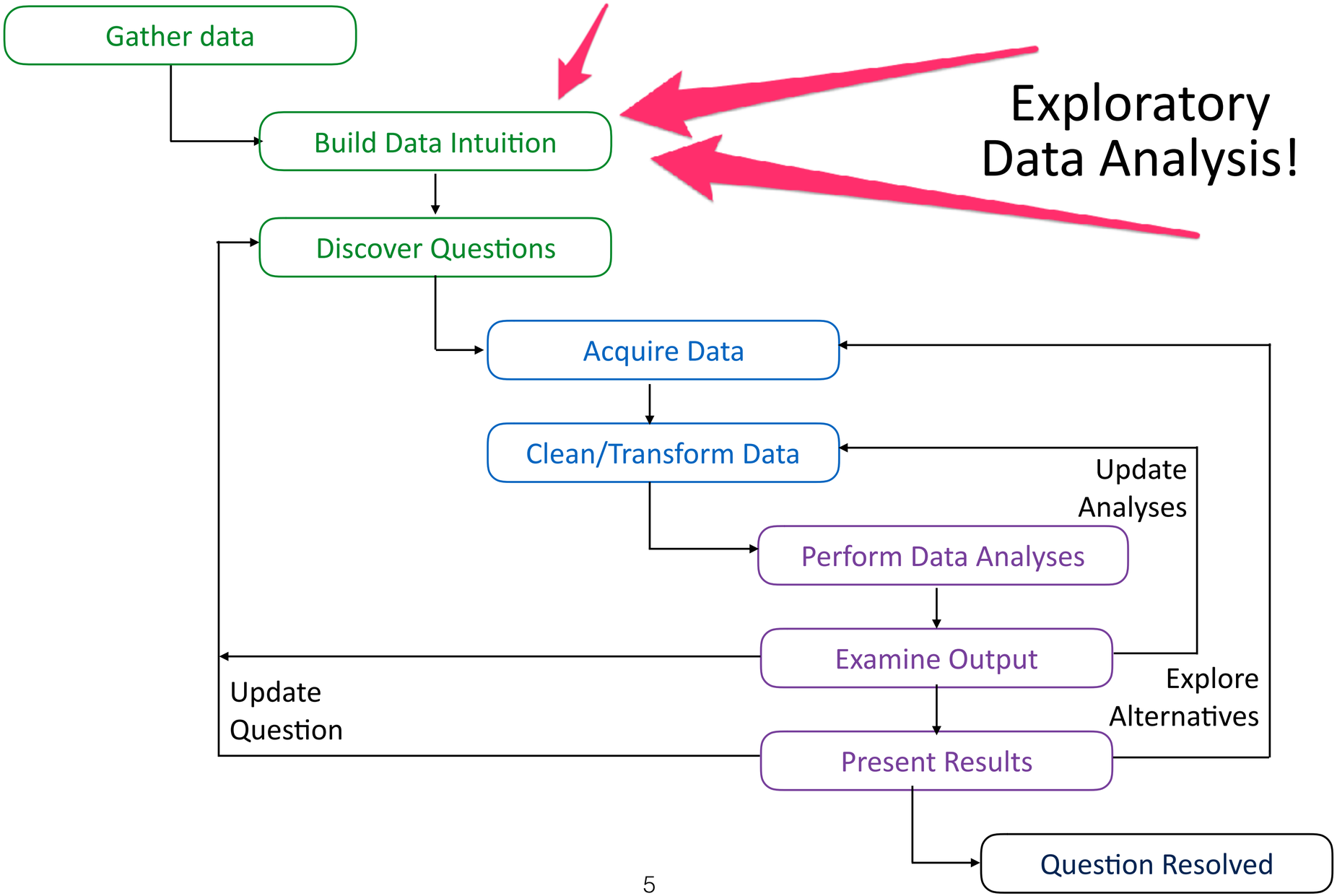
Research Workflow



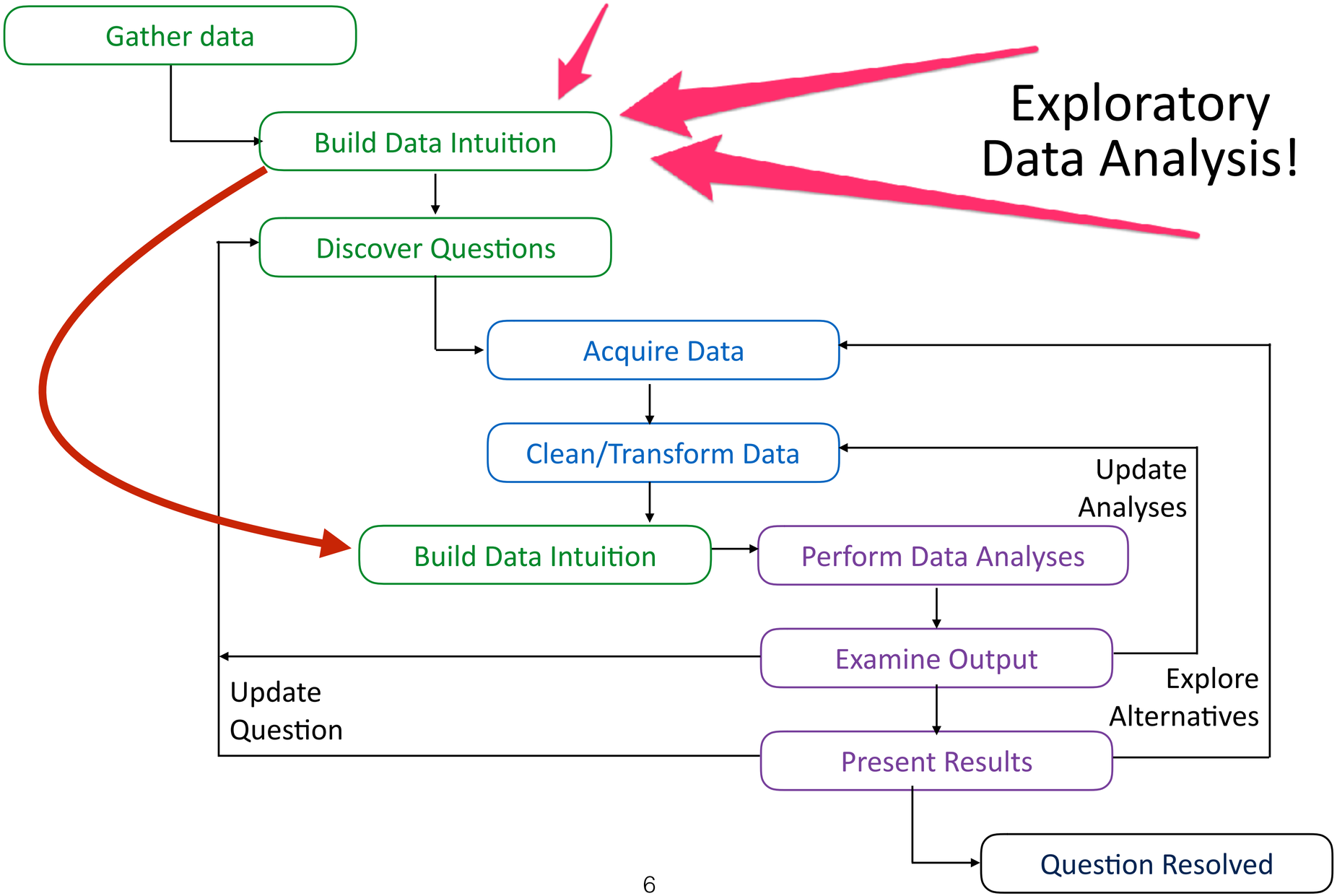
Reality of Research Workflow



Reality of Research Workflow

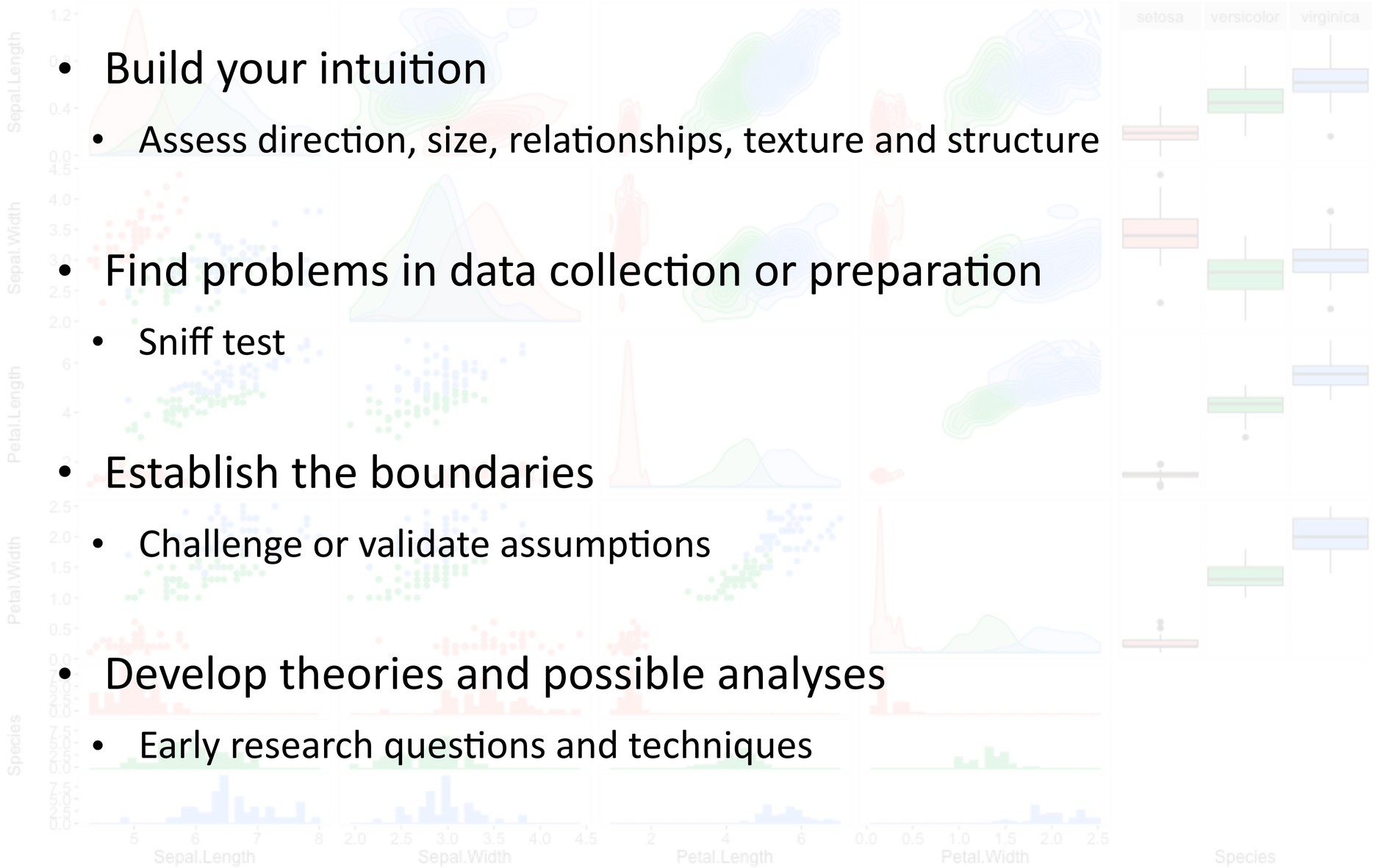


Reality of Research Workflow



Why EDA?

- Build your intuition
  - Assess direction, size, relationships, texture and structure
- Find problems in data collection or preparation
  - Sniff test
- Establish the boundaries
  - Challenge or validate assumptions
- Develop theories and possible analyses
  - Early research questions and techniques



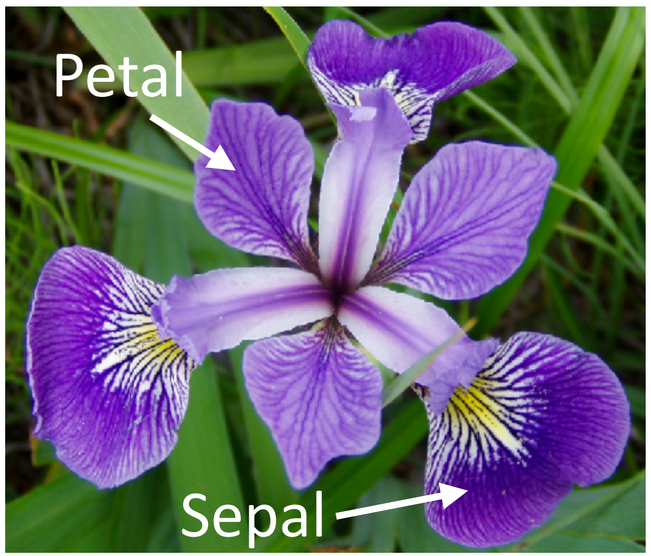
“Iris” data set

```
'data.frame':    150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100	setosa :50
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor:50
Median :5.800	Median :3.000	Median :4.350	Median :1.300	virginica :50
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199	
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500	

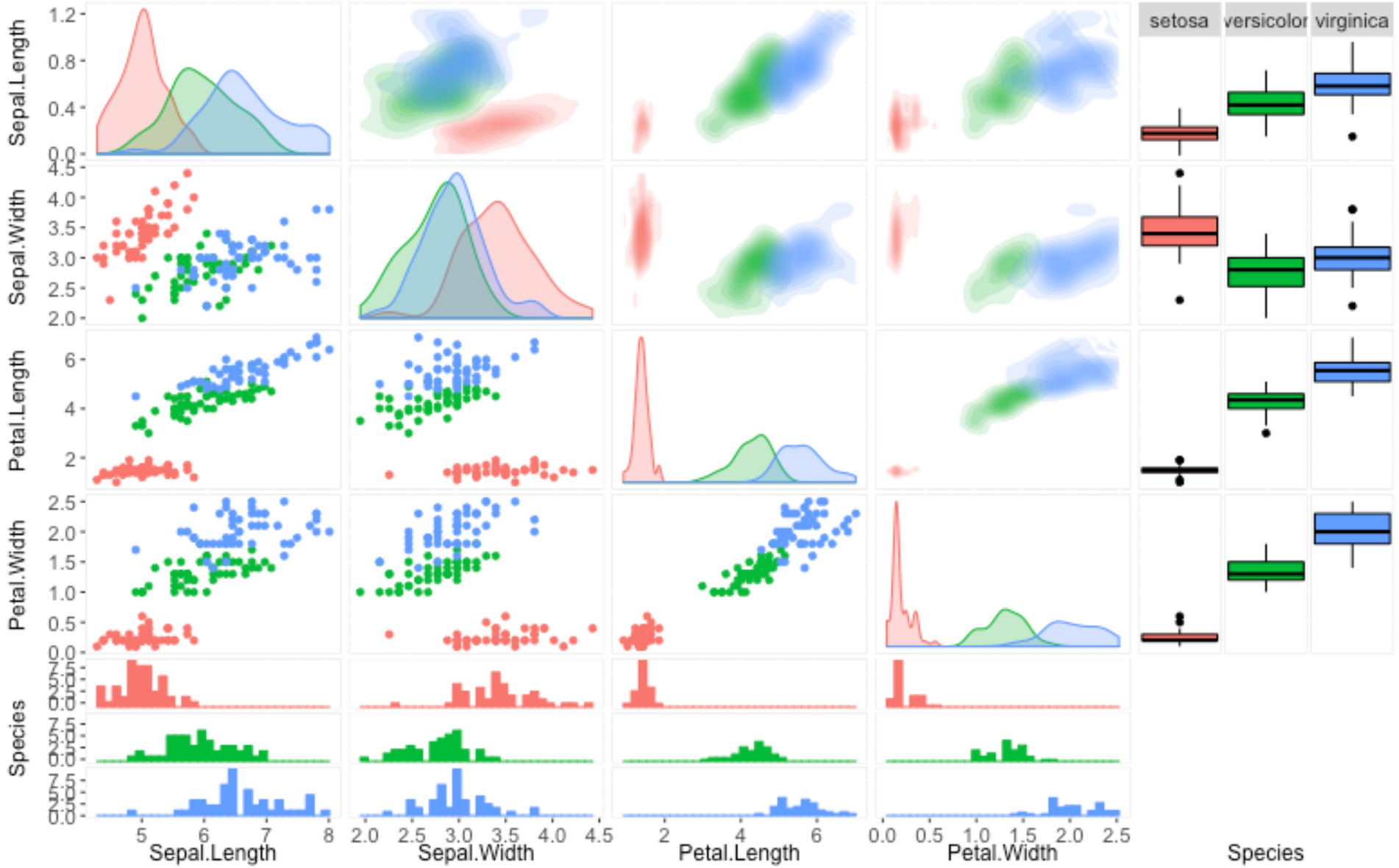
Iris Setosa

Iris Versicolor



Iris Virginica

Iris


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
setosa
versicolor
virginica
Species

Text Summaries

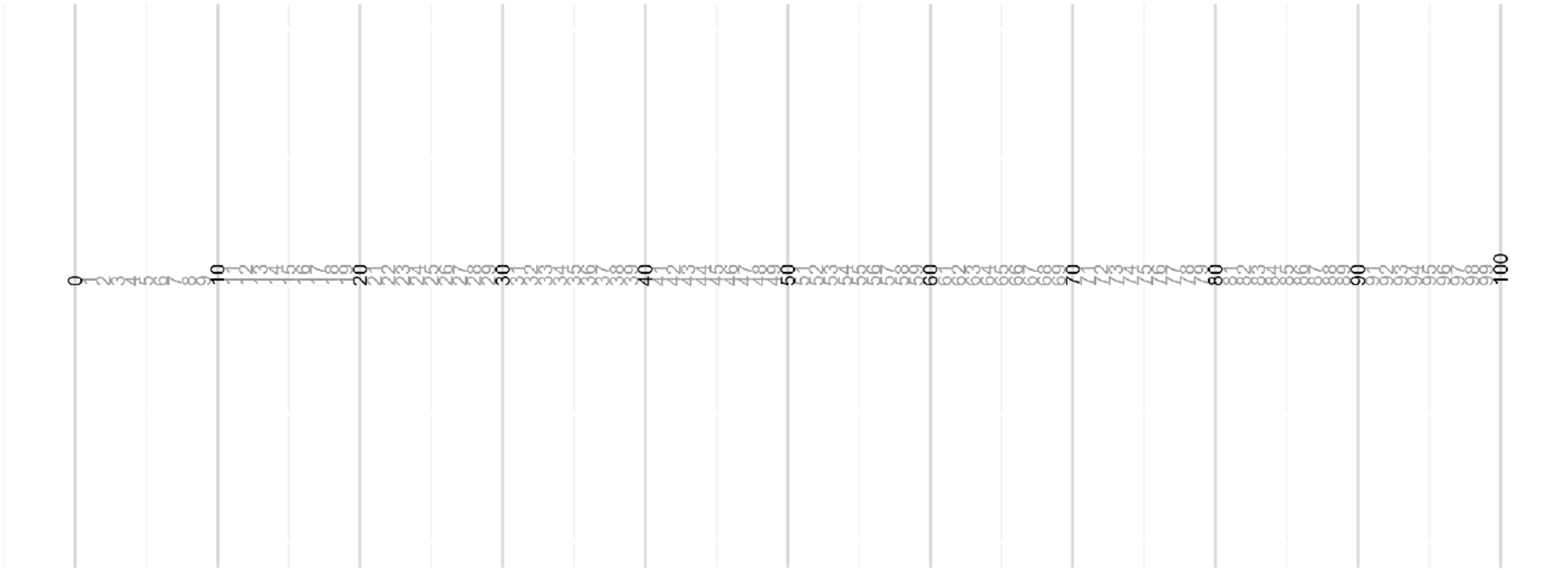
Python (pandas) `describe()`

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

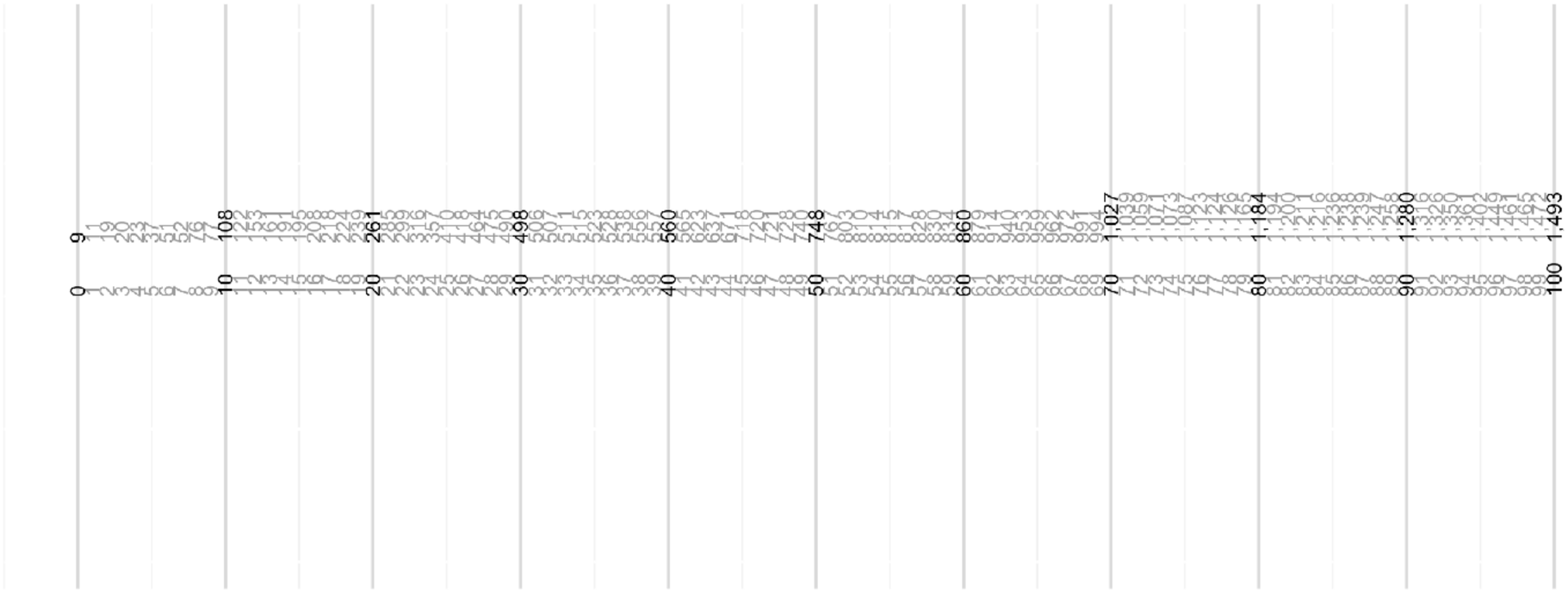
R `summary()`

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100	setosa :50
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor:50
Median :5.800	Median :3.000	Median :4.350	Median :1.300	virginica :50
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199	
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500	

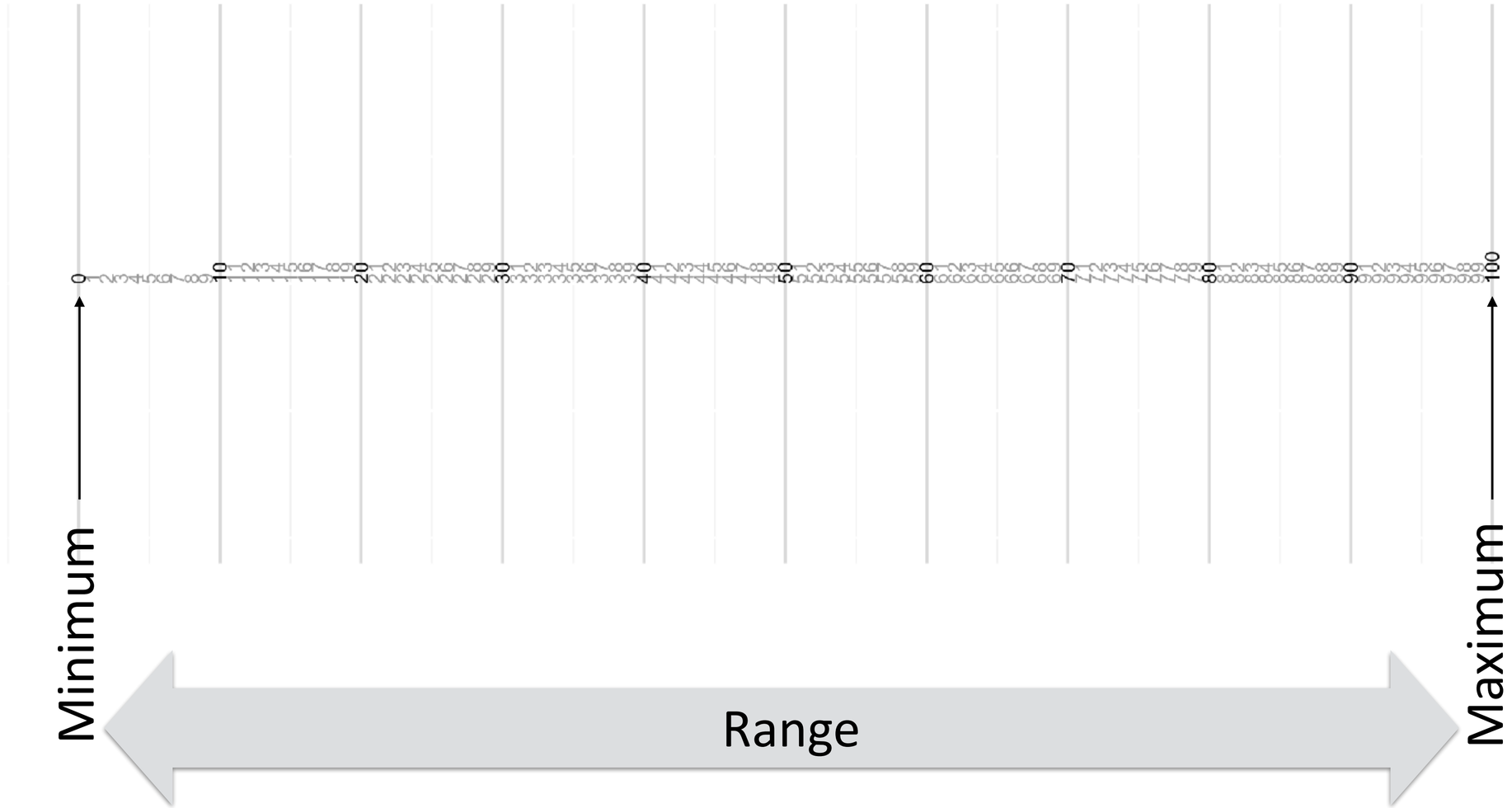
Single Variable



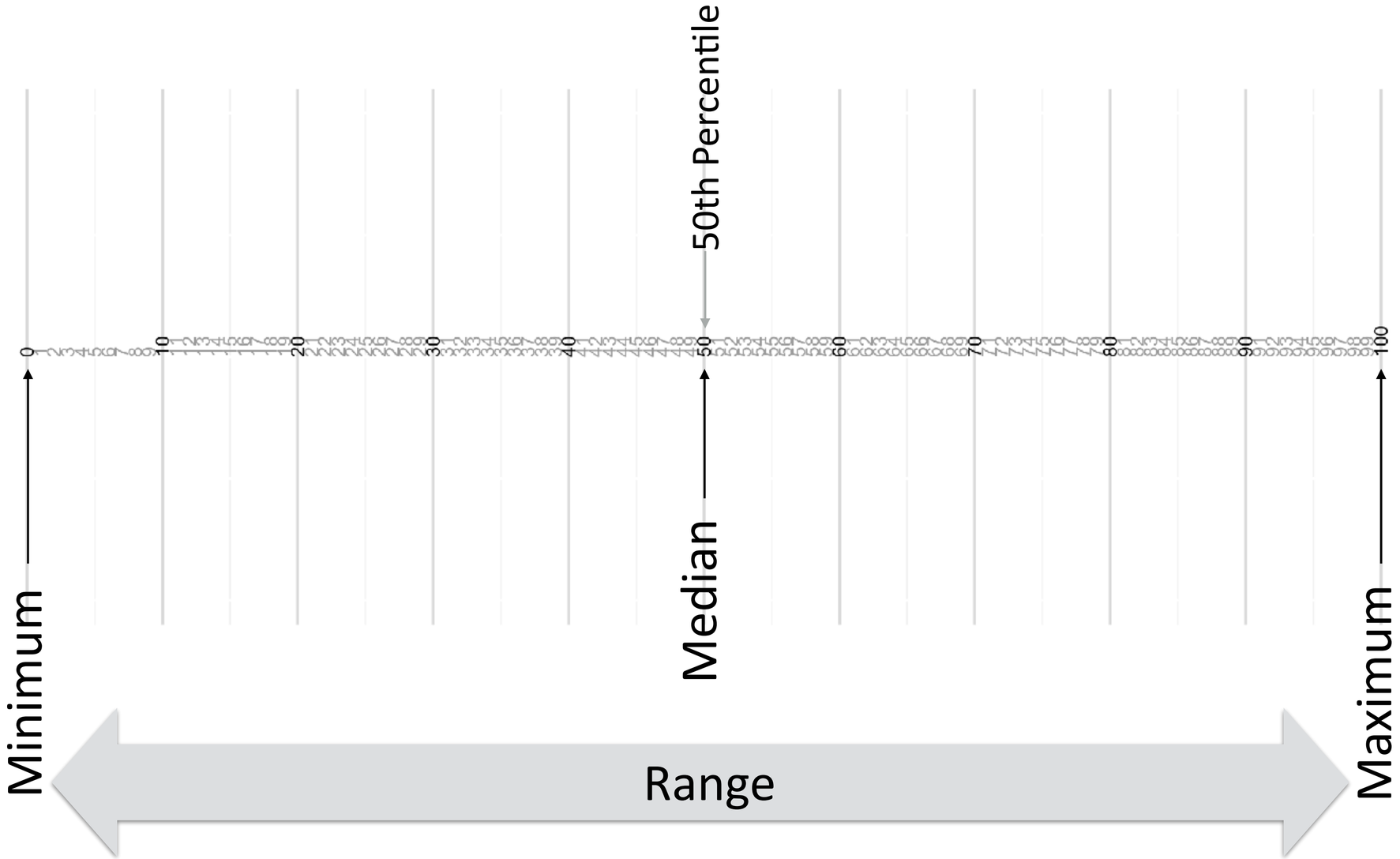
Single Variable



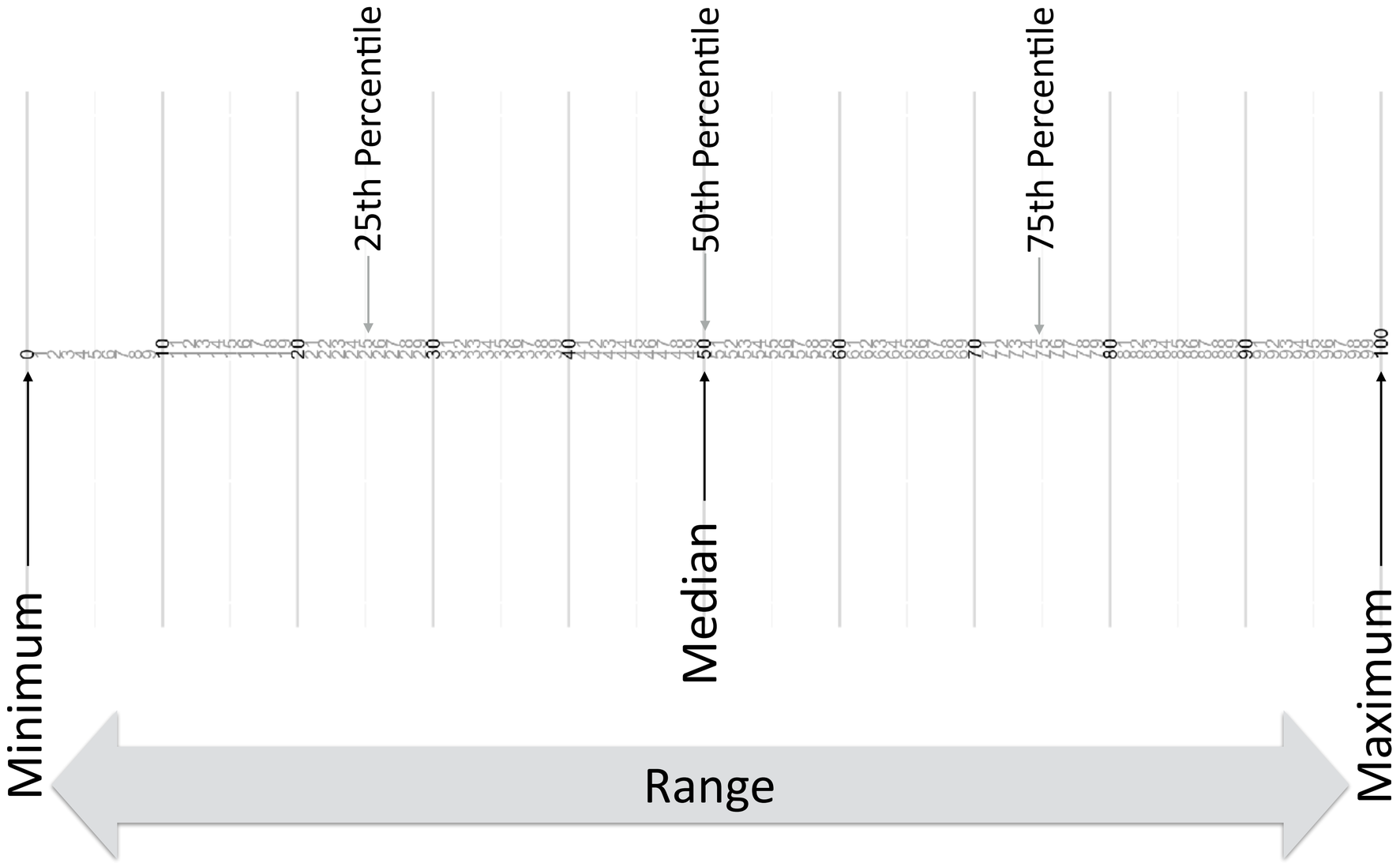
Single Variable



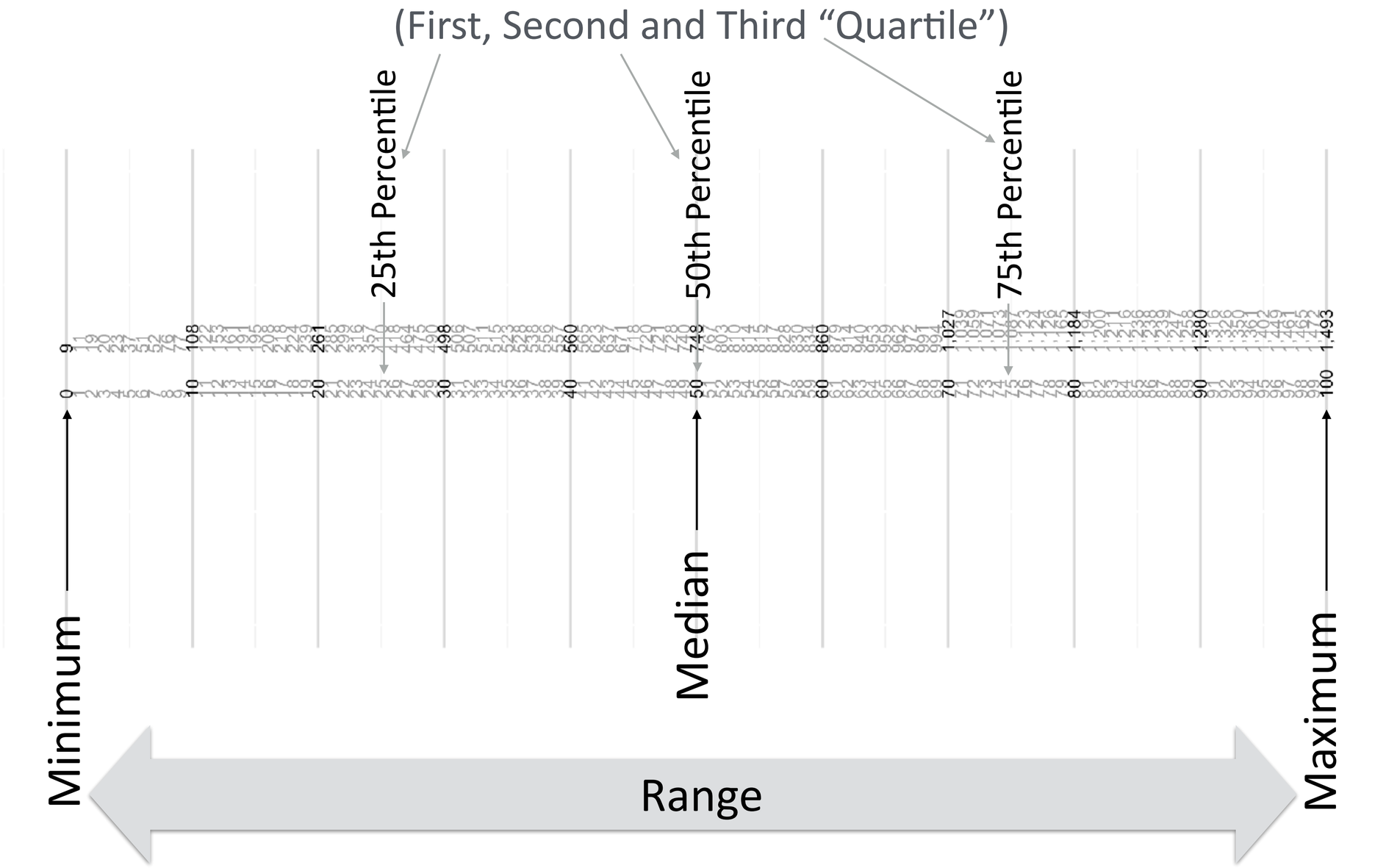
Single Variable



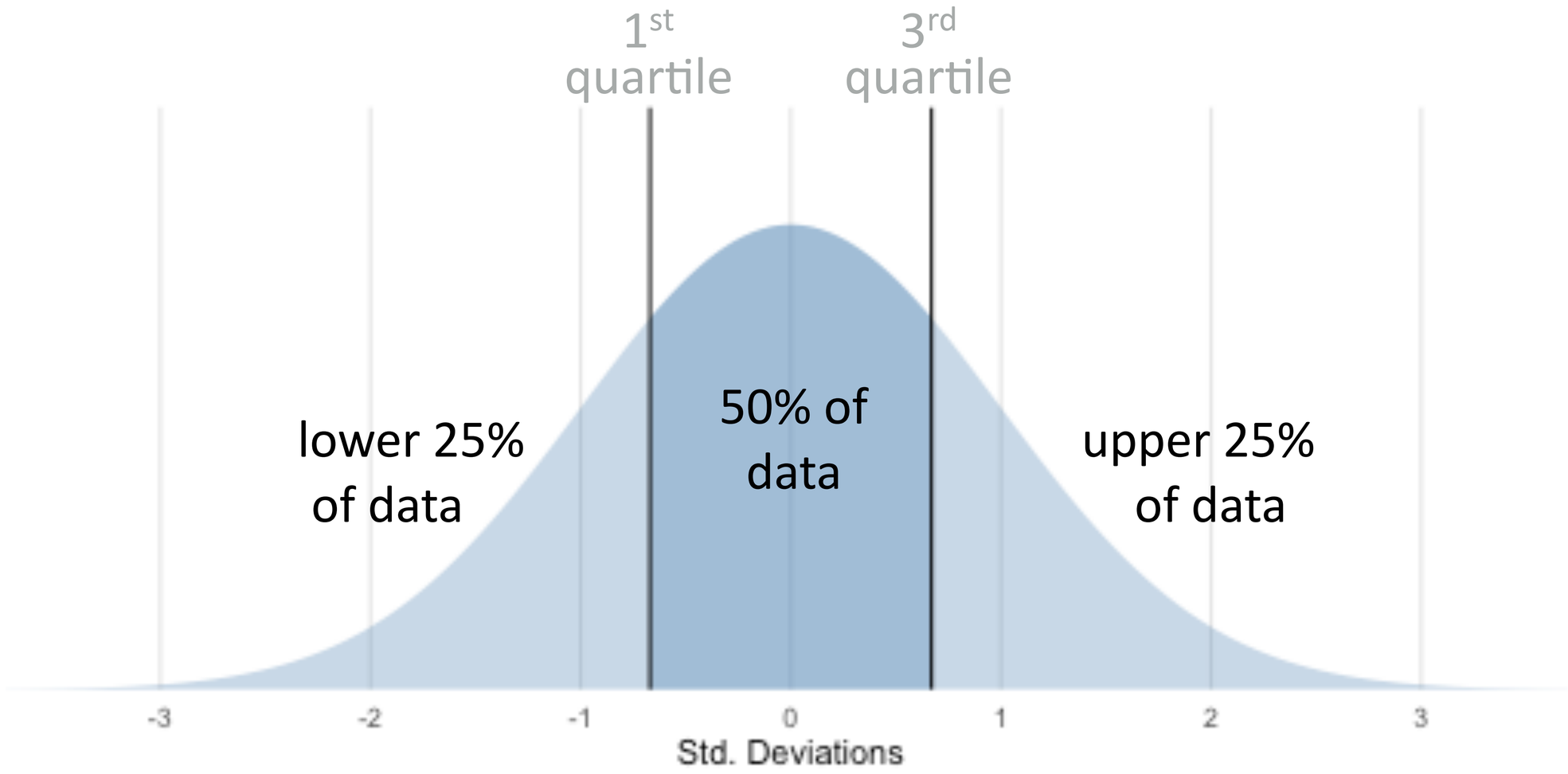
Single Variable



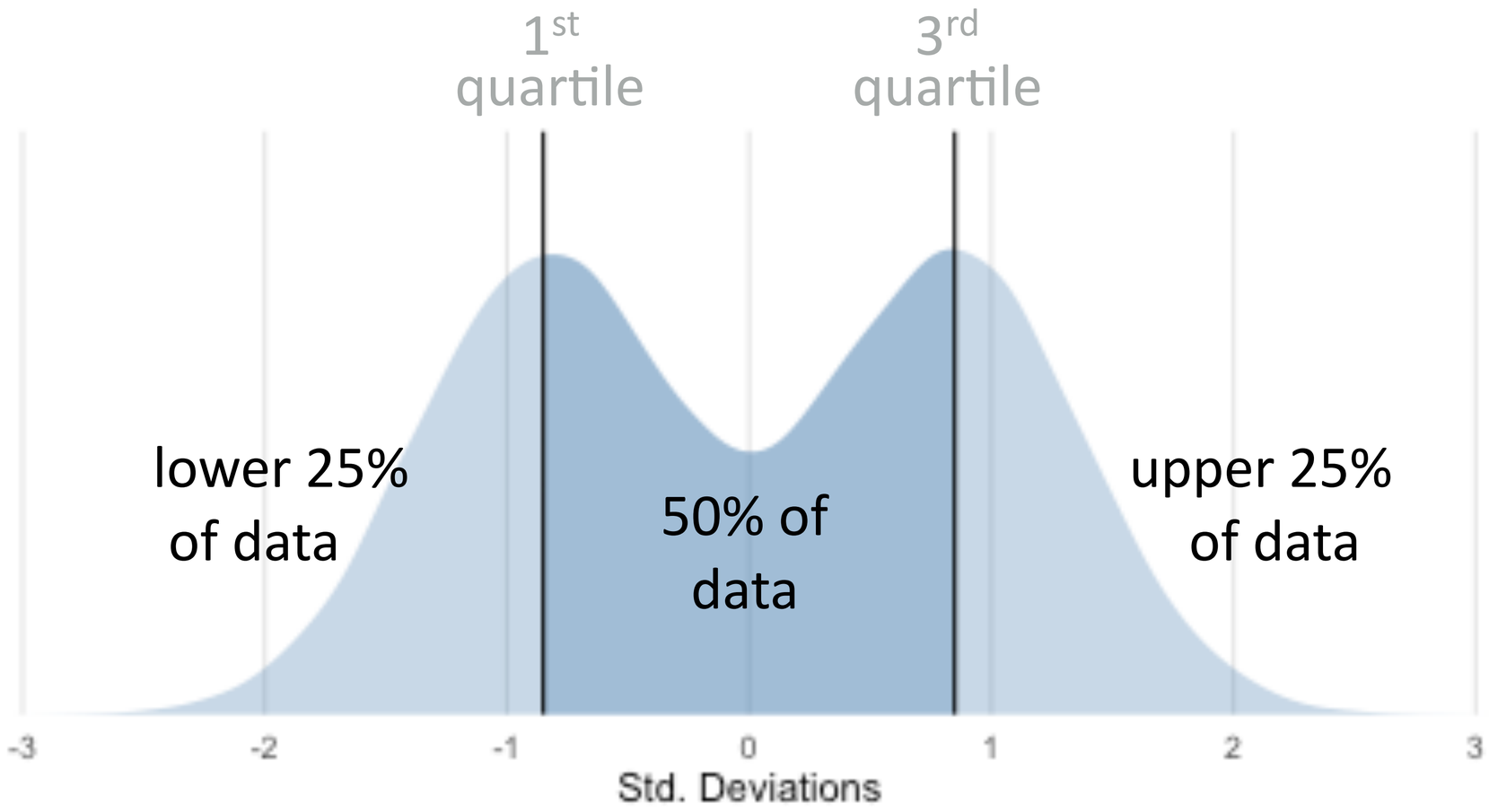
Five Number Summary



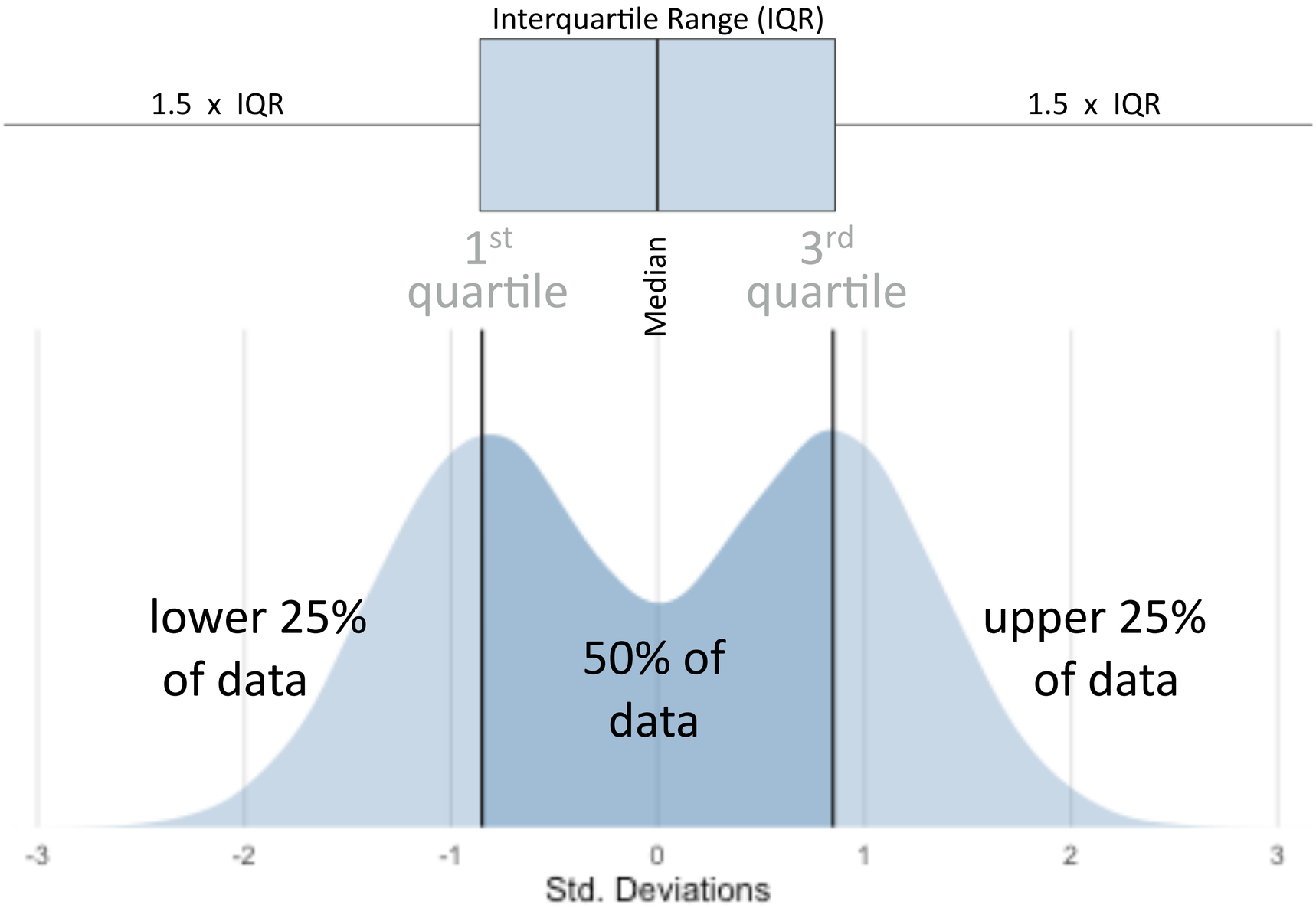
Visualizing 5-number summaries



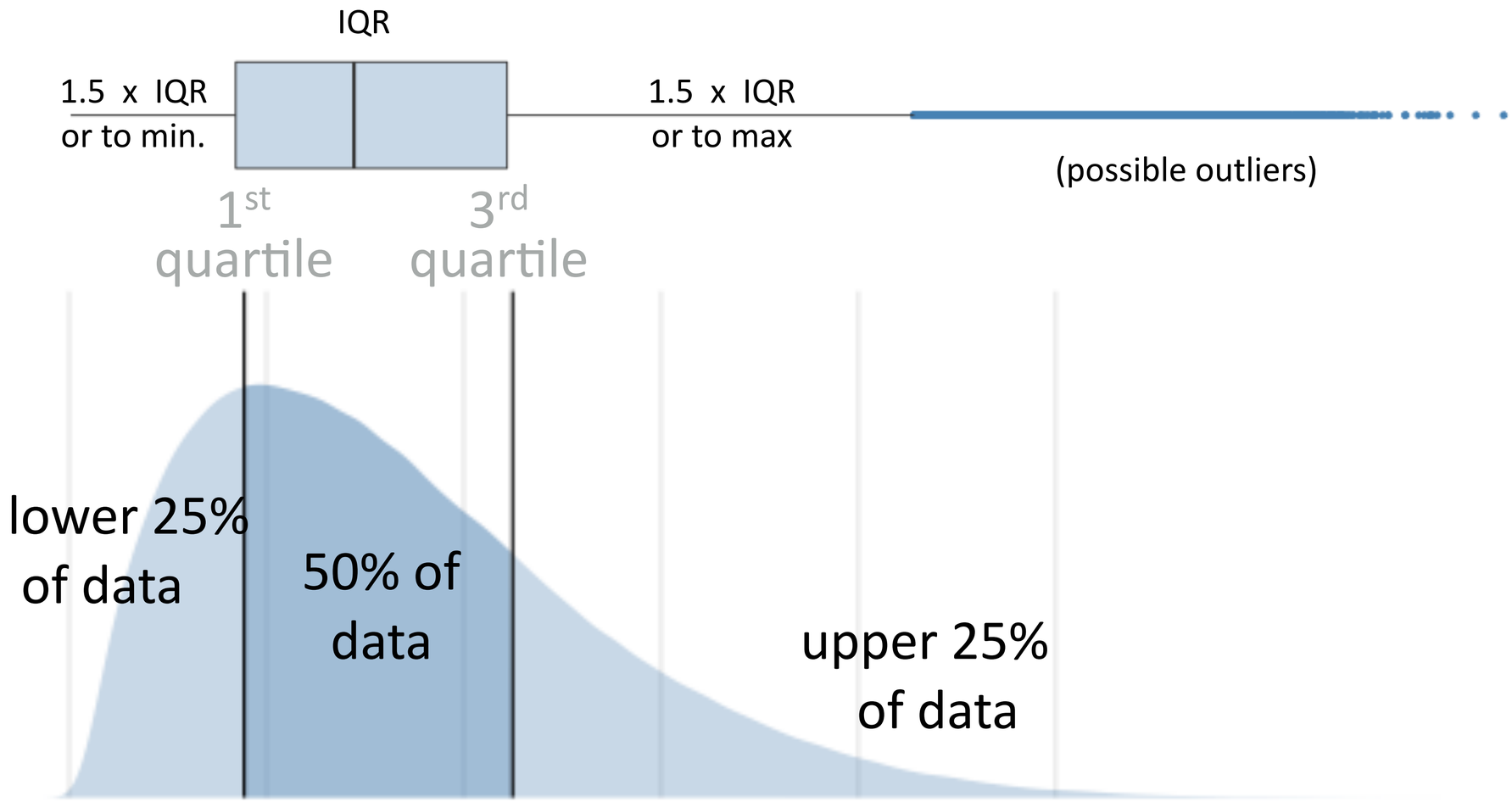
Bi-Modal (two modes/peaks)



Bi-Modal (two modes/peaks)

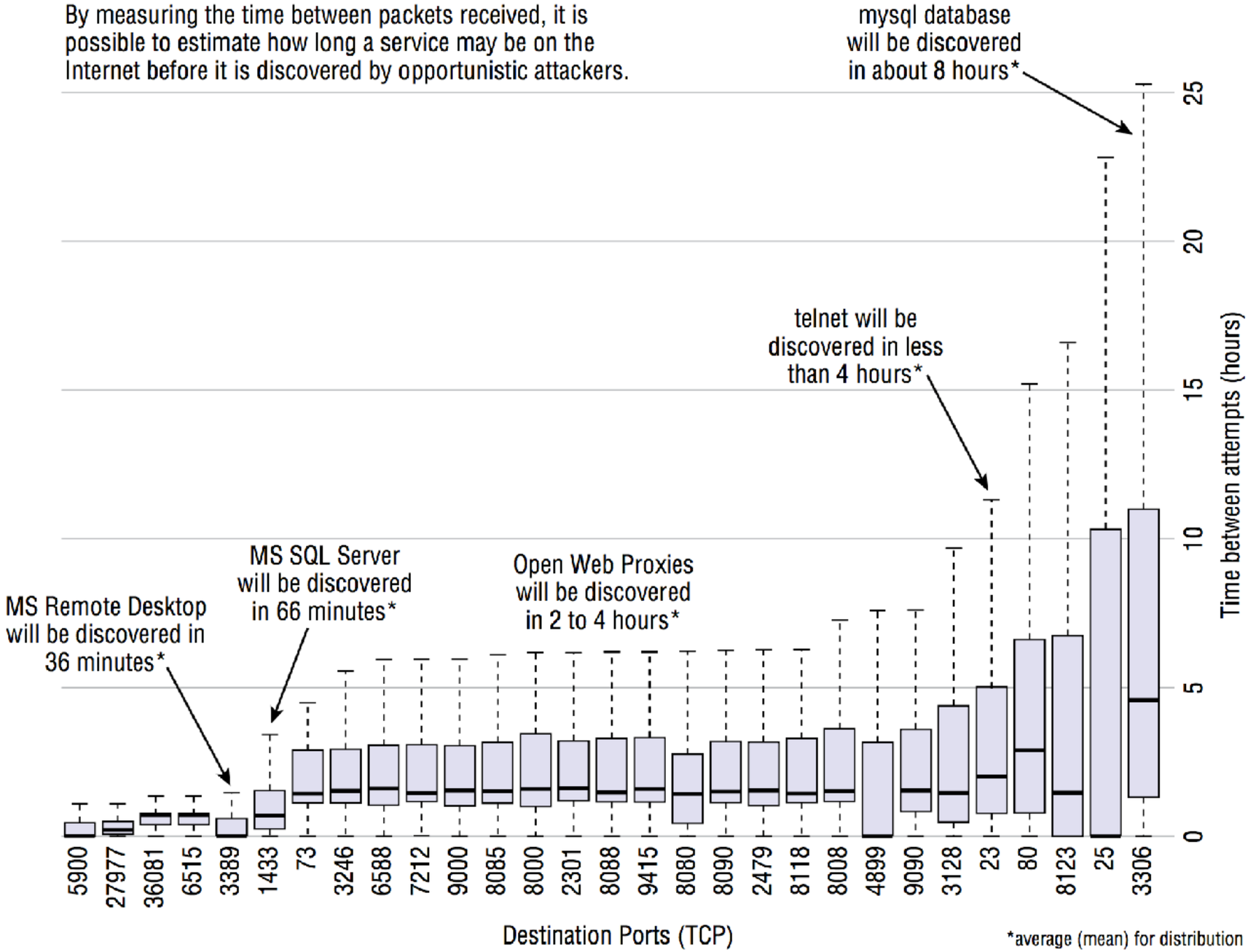


5-Number Summaries with Skew

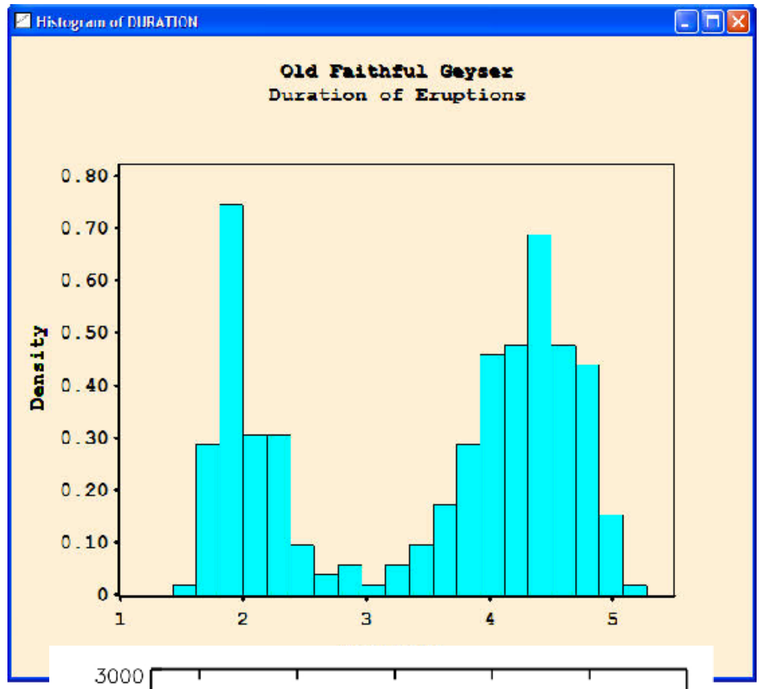


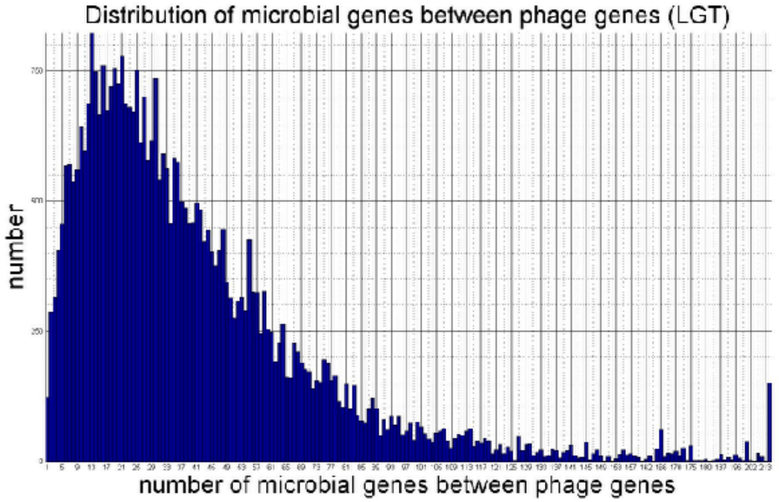
How long will a service go undiscovered by opportunistic attackers?

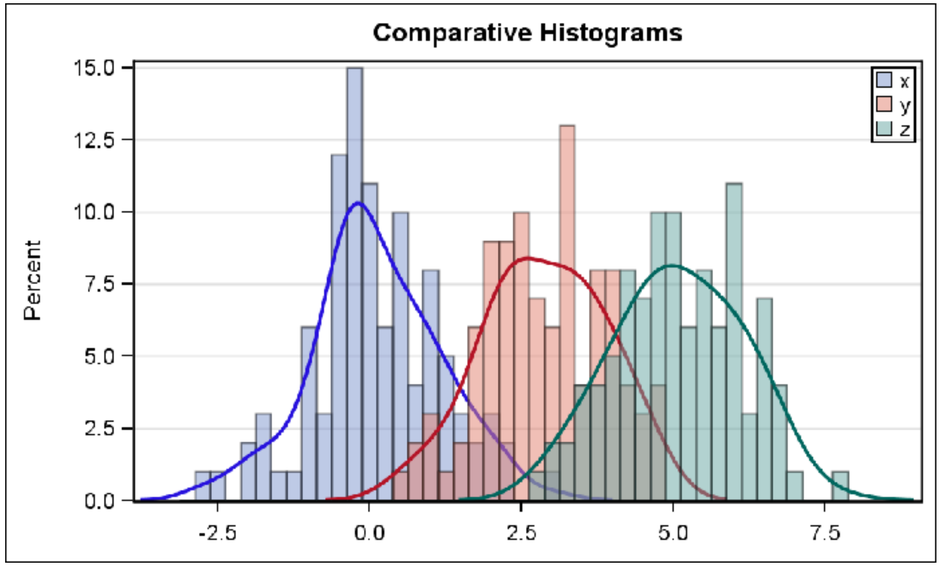
By measuring the time between packets received, it is possible to estimate how long a service may be on the Internet before it is discovered by opportunistic attackers.

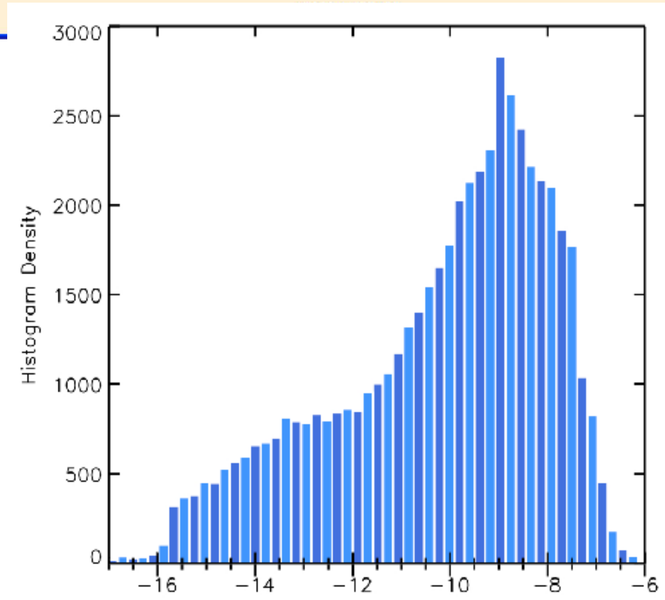


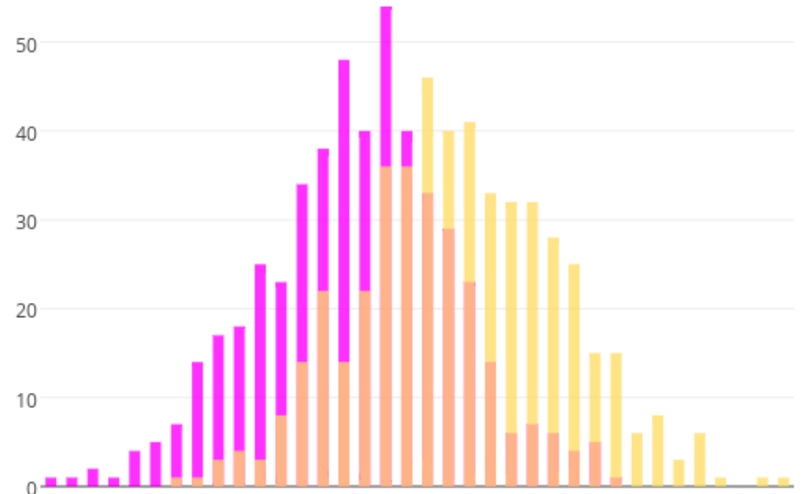
Visualizing Distributions: Histogram

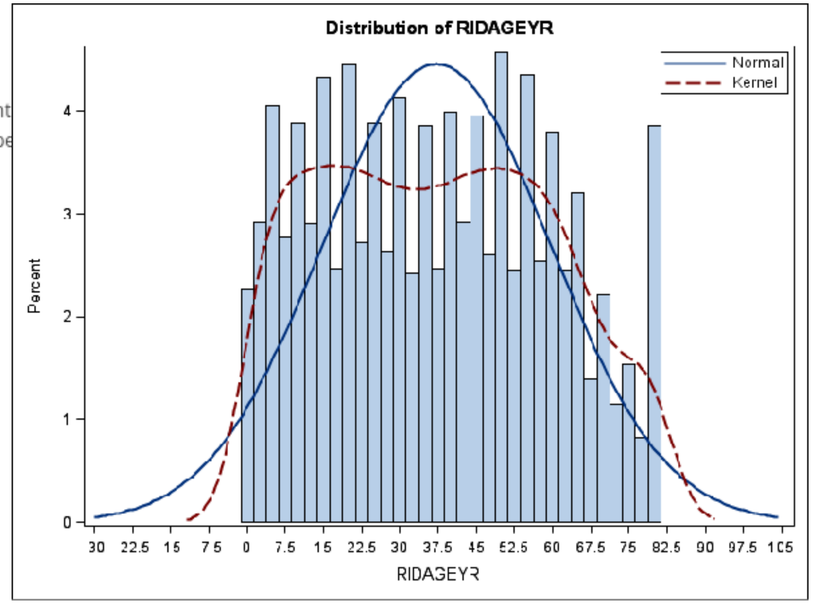




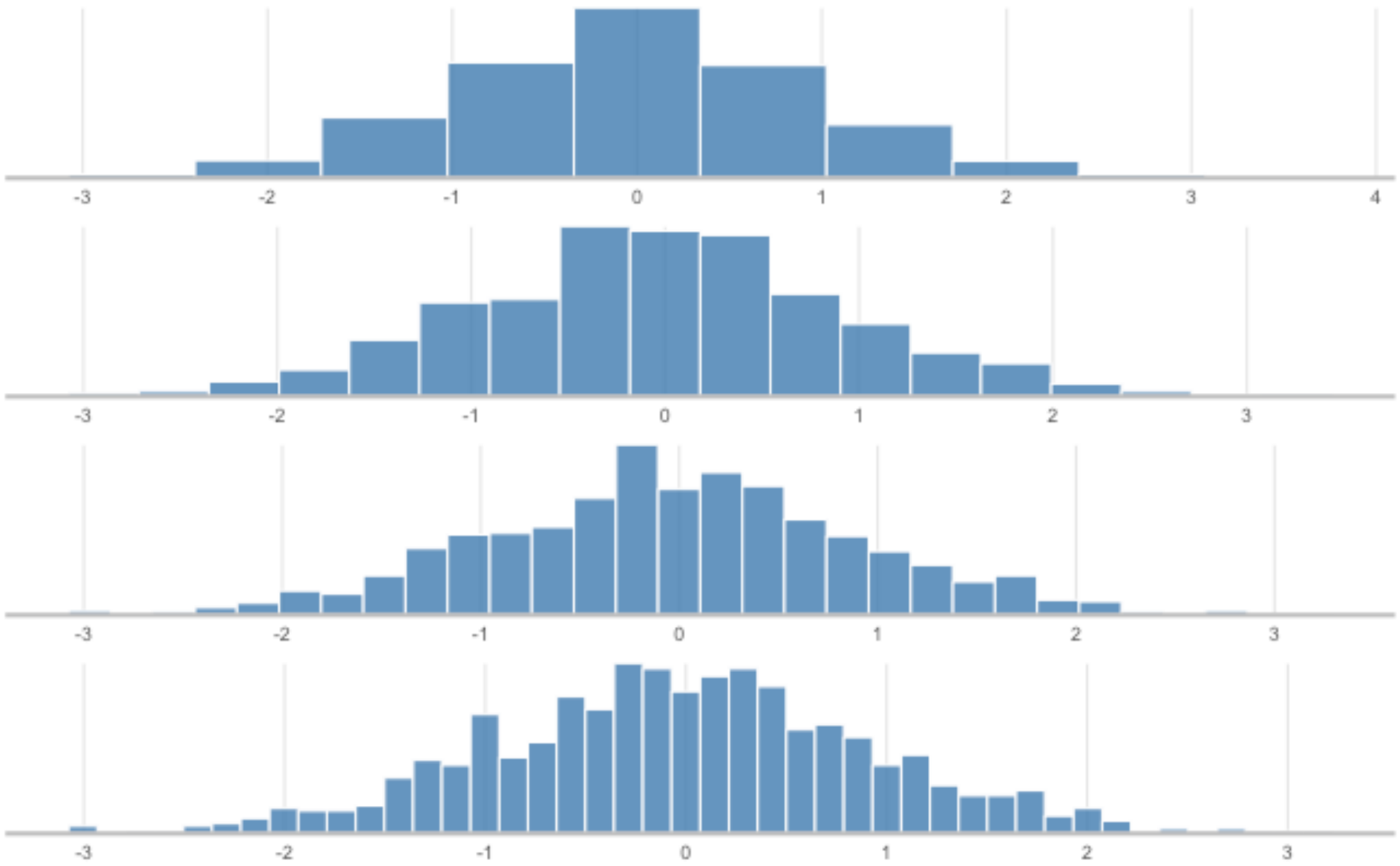




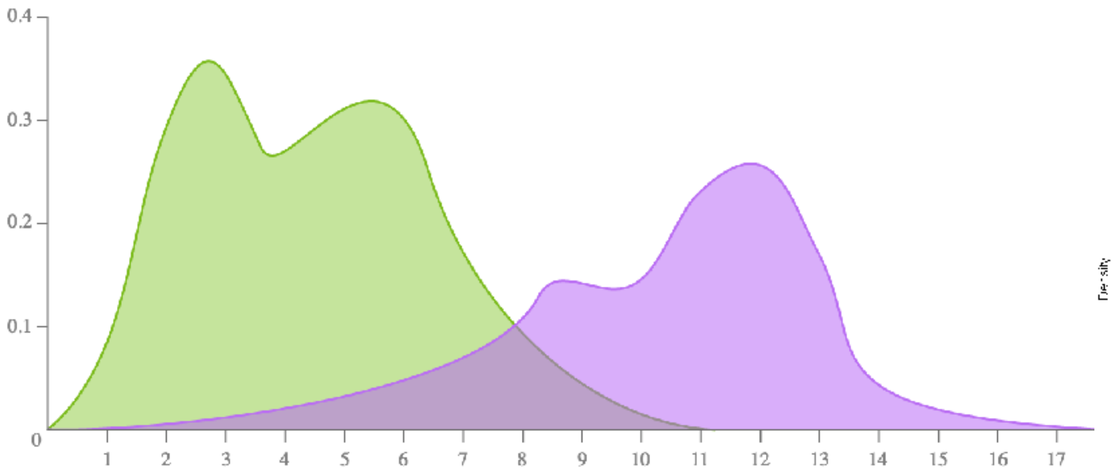


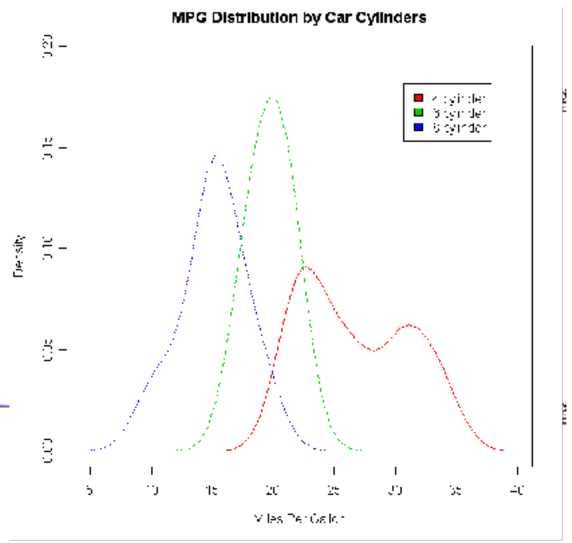


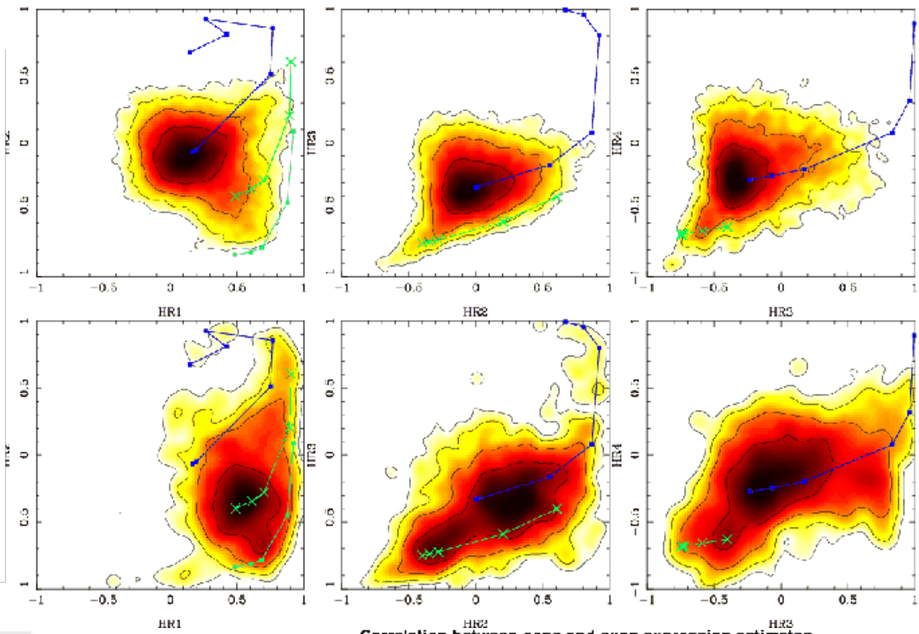
Visualizing Distributions: Histogram

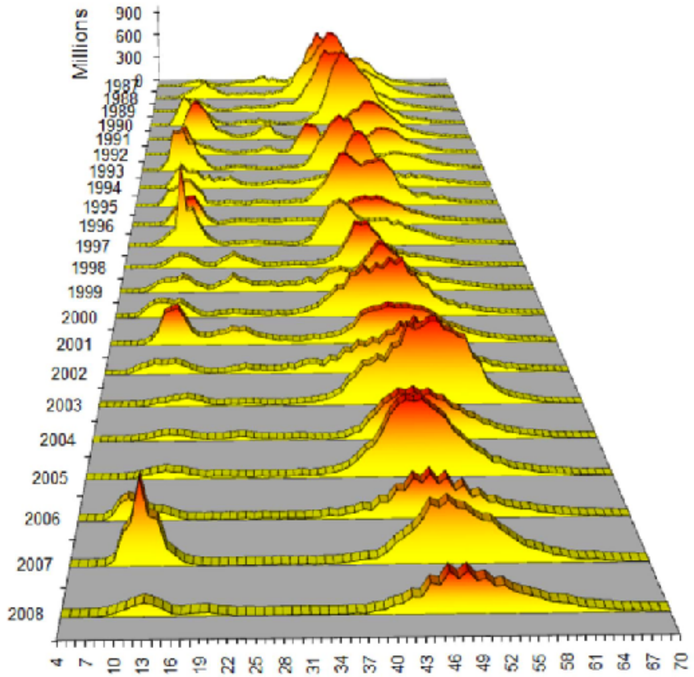


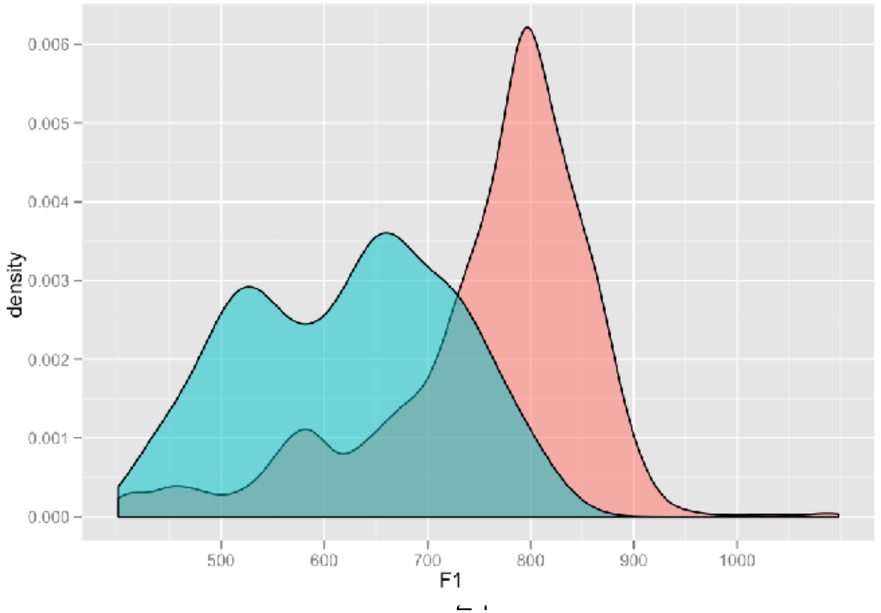
Visualizing Distributions: Density plots

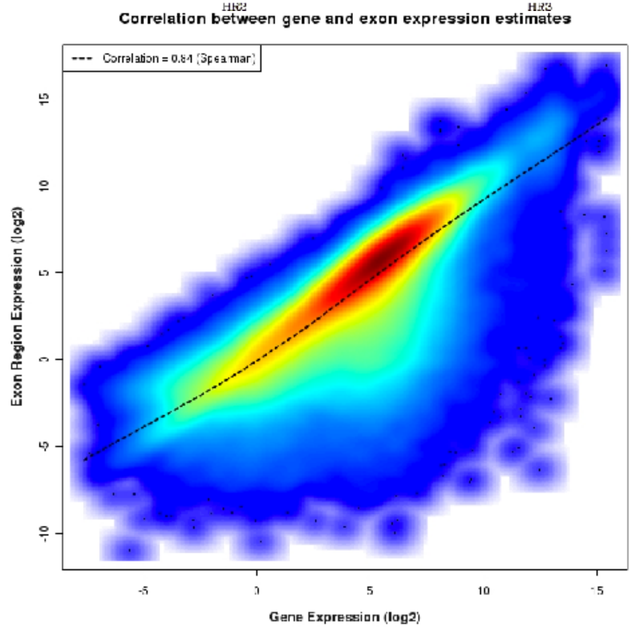




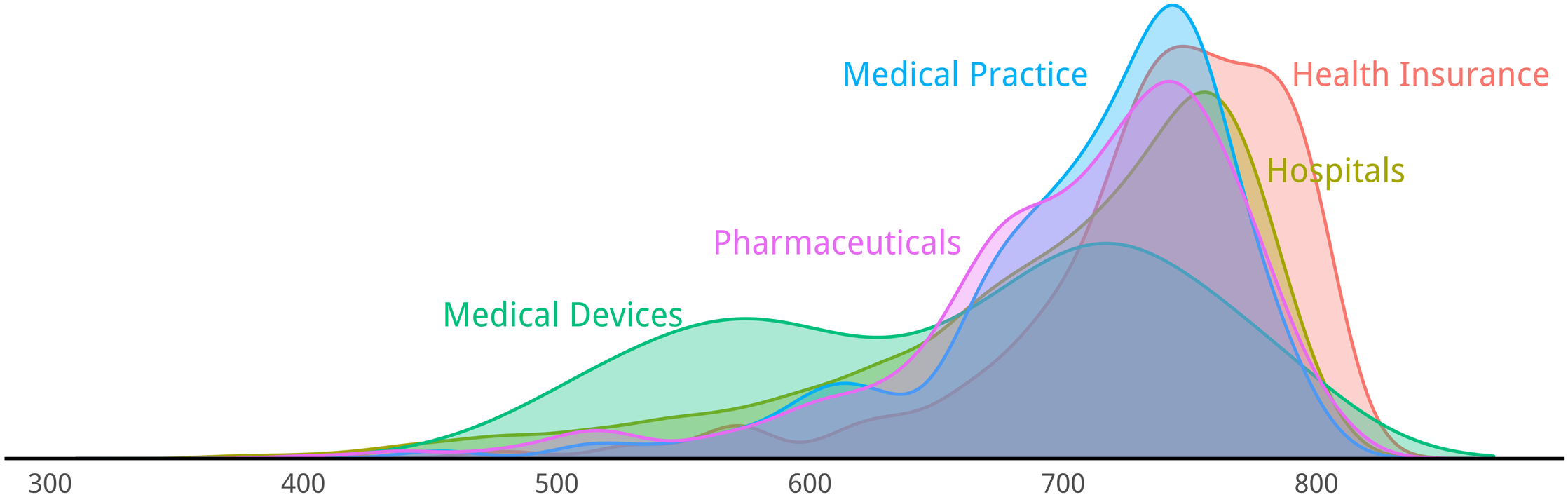






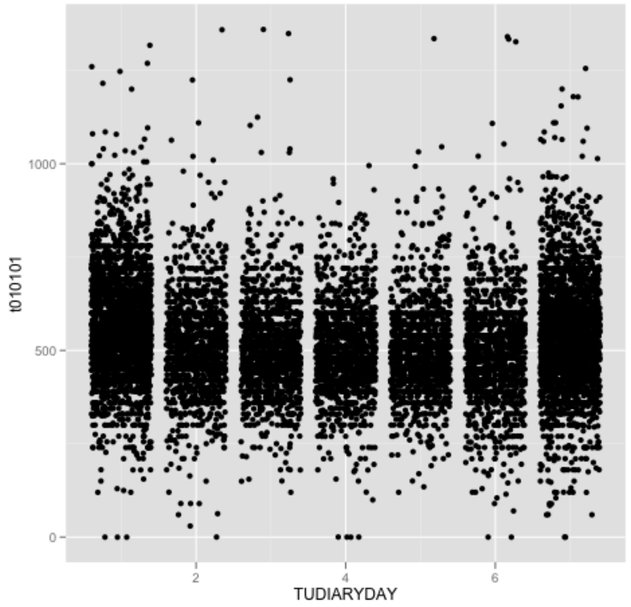


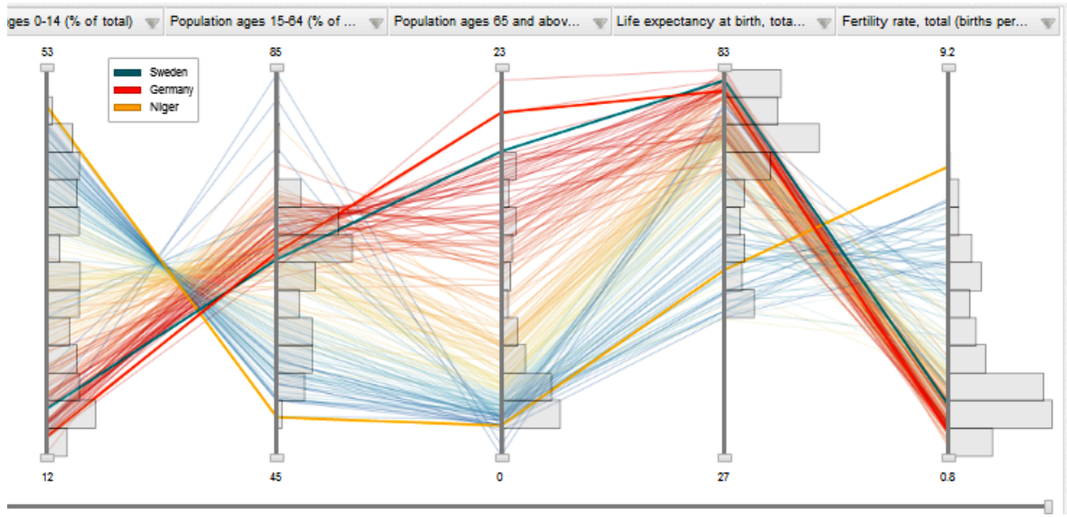
Visualizing Distributions: Density plots

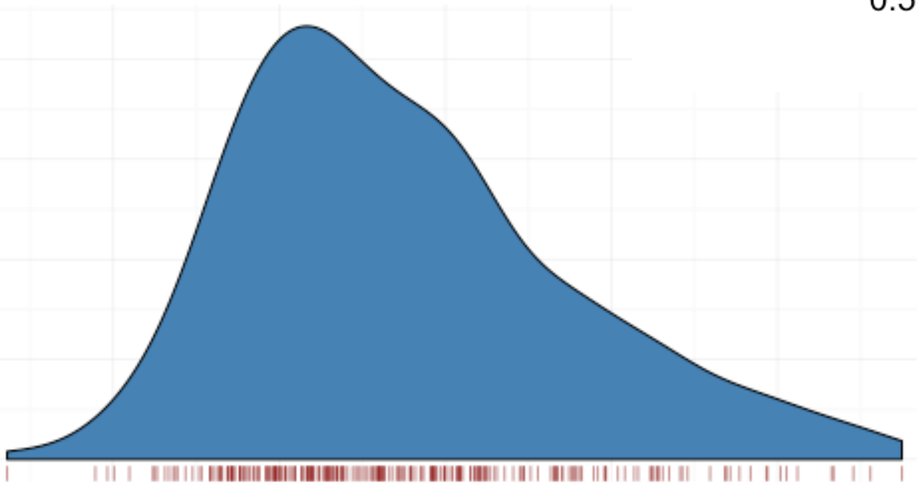


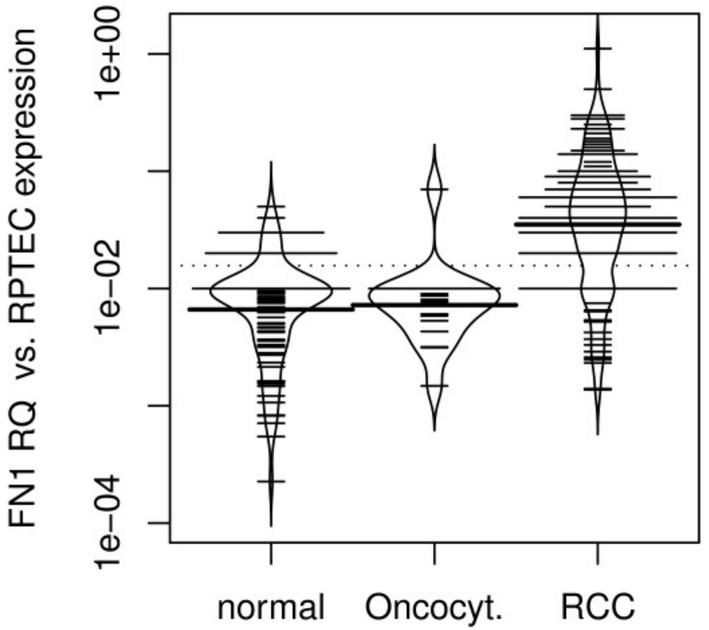
- Great for comparing different sized samples
- Nobody understands them

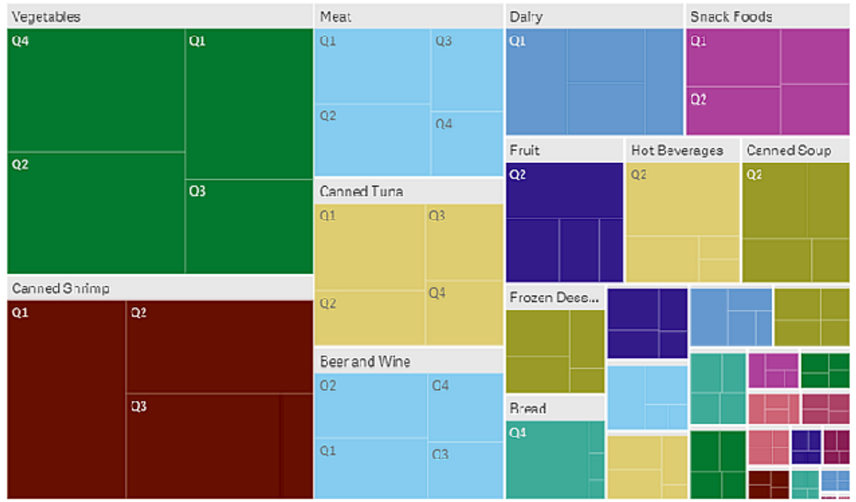
Visualizing Distributions: Others





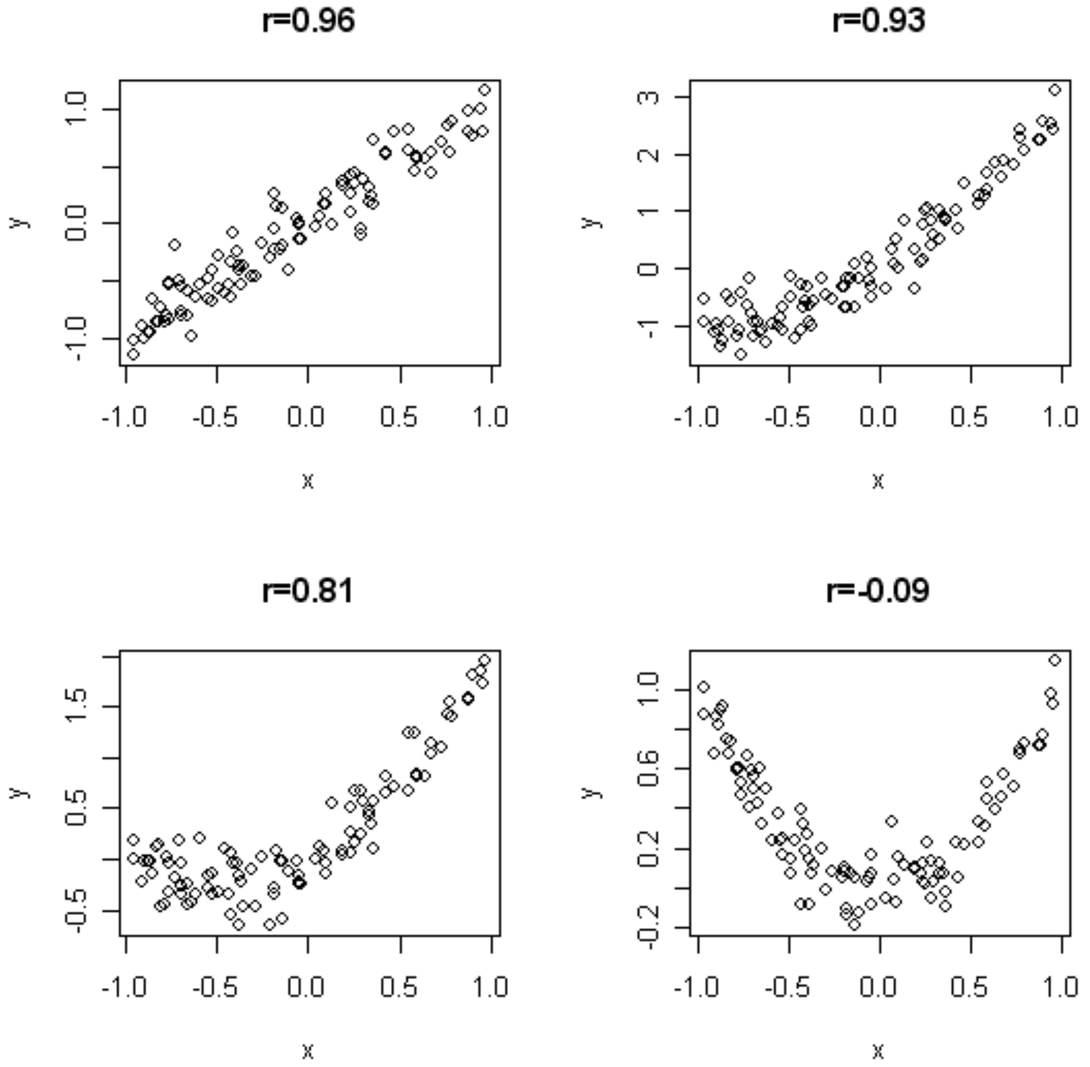




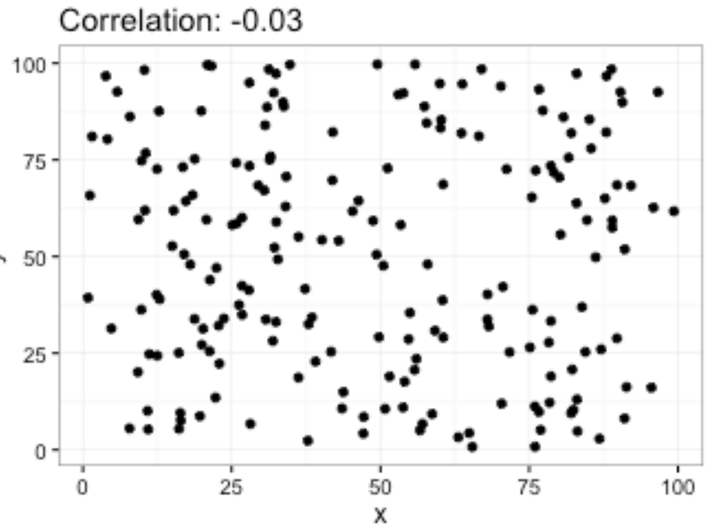


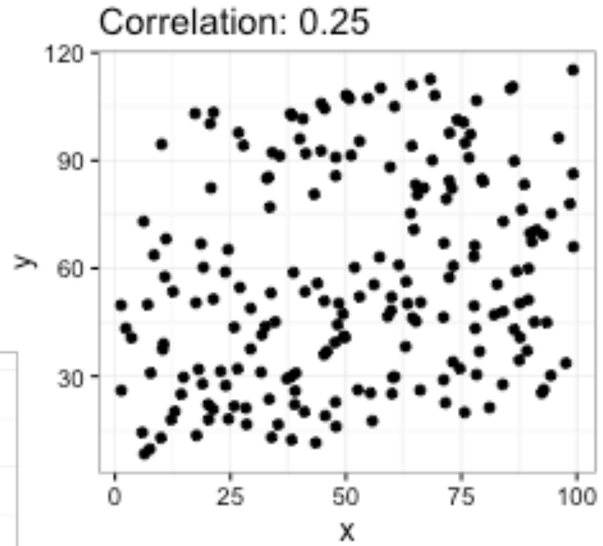
Correlations

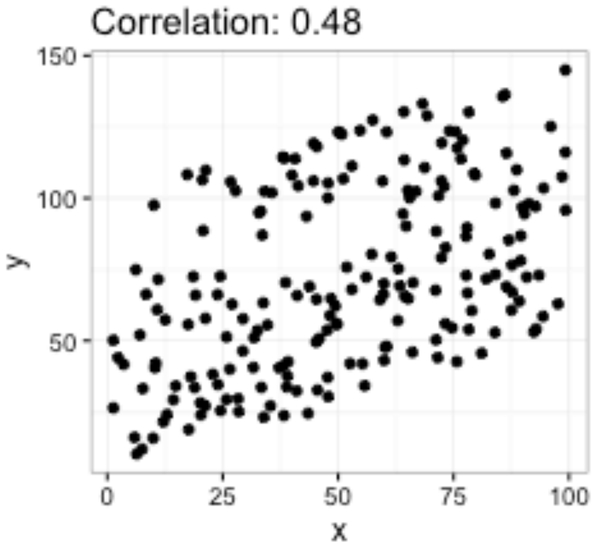
- Measure of the relationship between two continuous variables
- Output metric is between -1 and 1:
  - 1 is perfect positive correlation
  - 0 is no correlation
  - −1 is perfect negative correlation
- Most common is Pearson's Correlation
  - Typically labelled as "r"
- Two Statistical Tests
  - Test if the correlation is non-zero
  - Confidence of true relationship

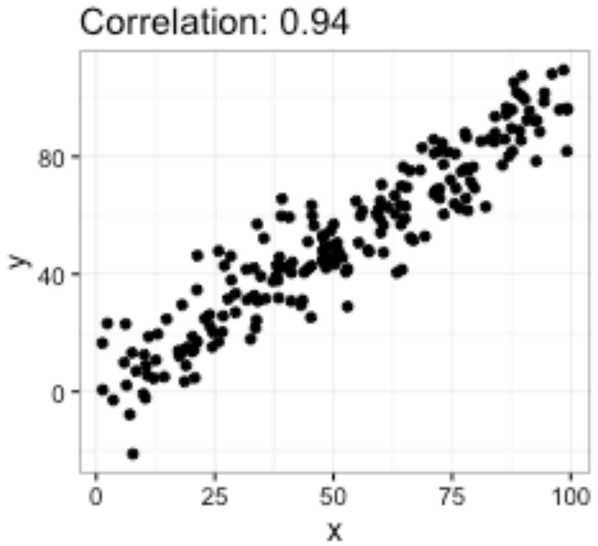


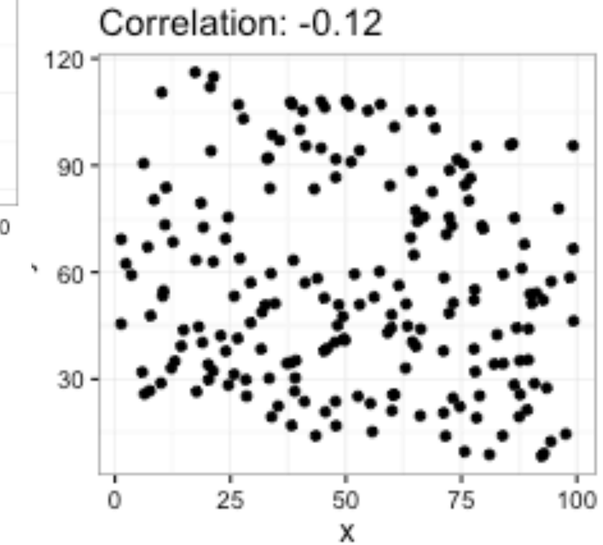
Correlations

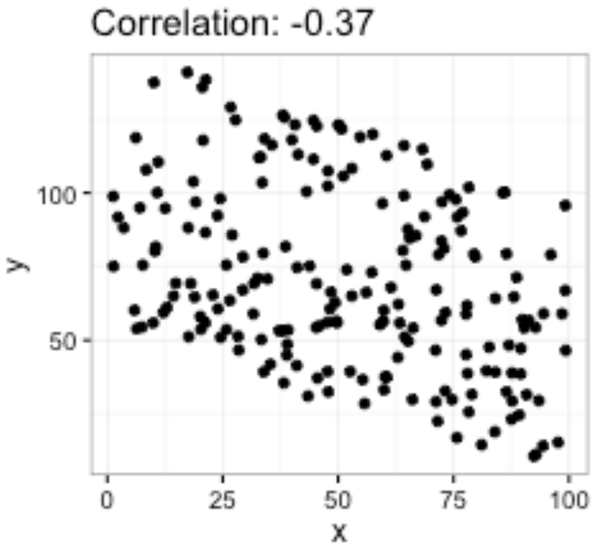


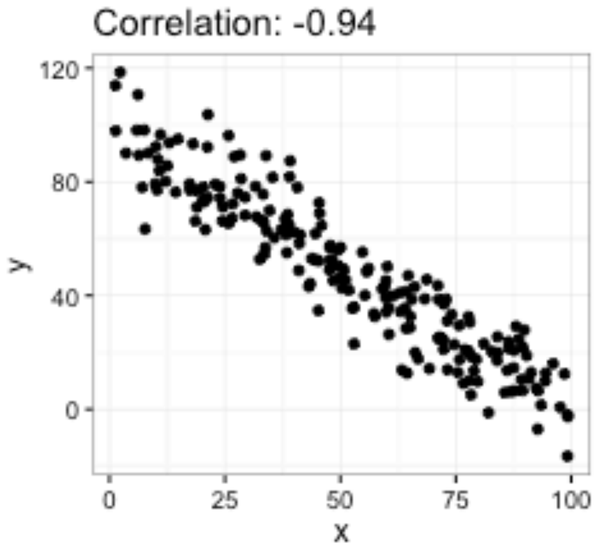








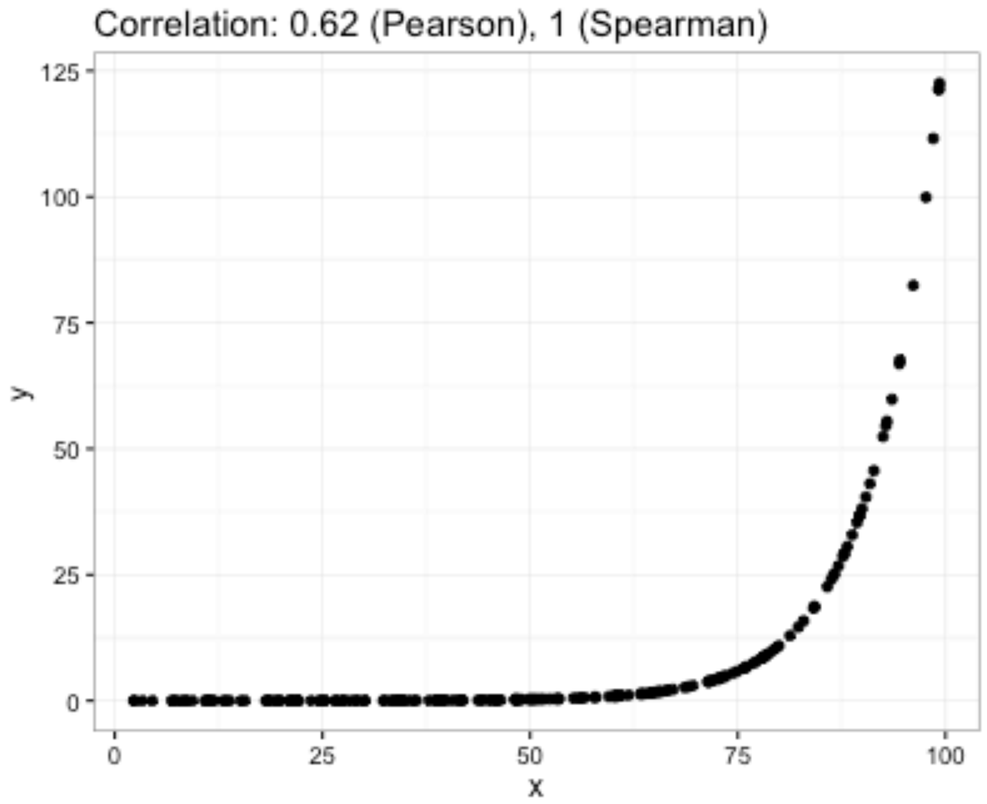


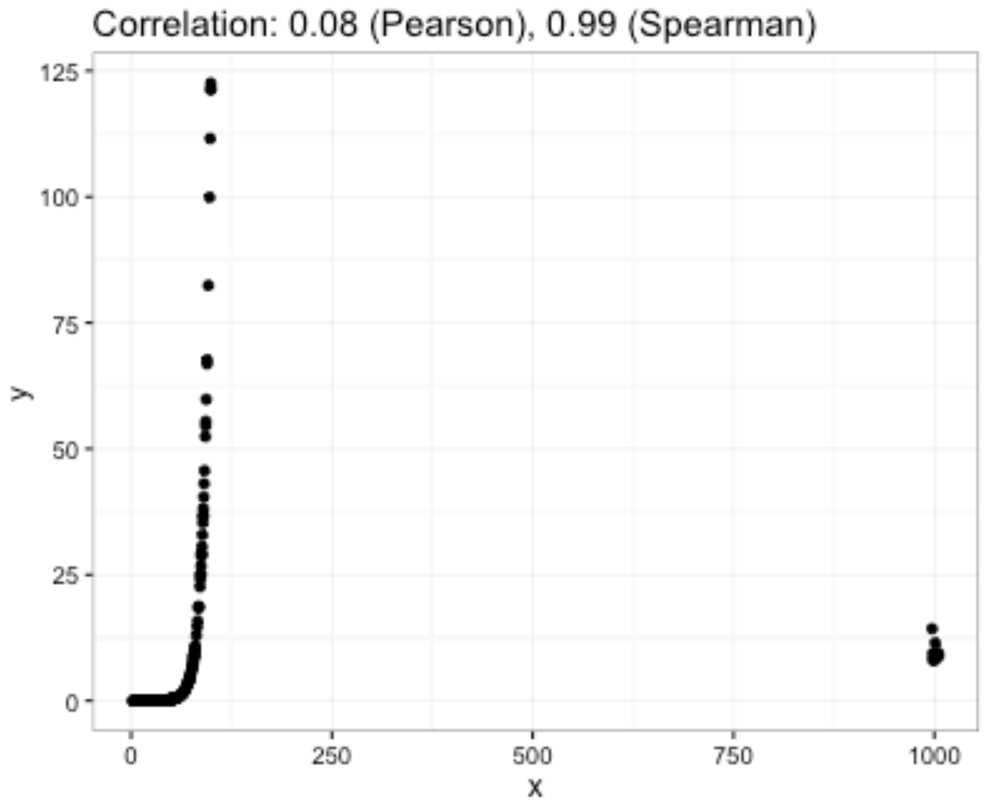


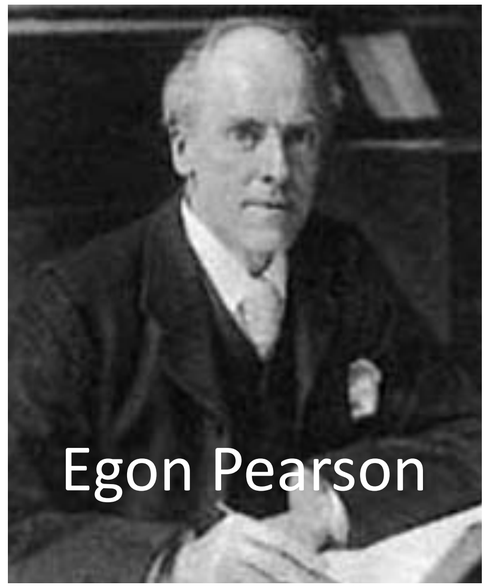
Try guessing at:
http://guessthecorrelation.com/

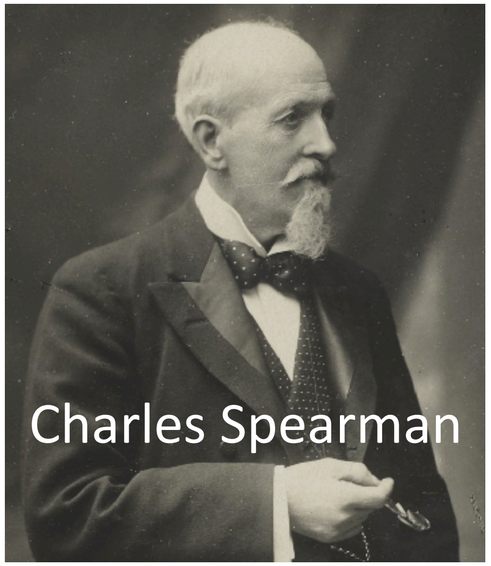
Correlations

- “Pearson” correlation works well on linear relationships with normal or uniform data
- “Spearman” calculates correlation on ranking of values
  - Works on non-linear relationship
  - Tolerates outliers quite well









Hypothesis Testing: p-value

Case Study:

Suppose you are tracking how much time employees spend connected to the VPN per week.

Assuming all others factors remain constant, when would you start to suspect an account may be compromised?

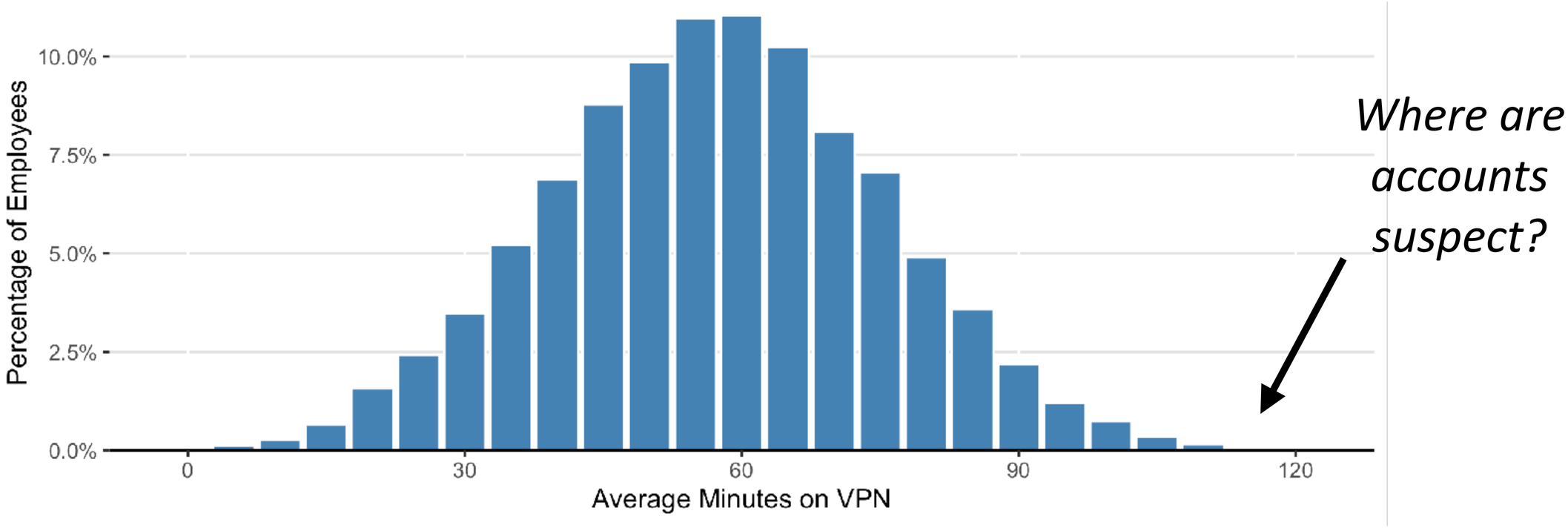
what if the account is above the 95th percentile of all users? …above 99th percentile?

Hypothesis Testing: p-value

Case Study:

Suppose you are tracking the average time each employee spends connected to the VPN per week.

Assuming all others factors remain constant, when would you suspect an account may be compromised?

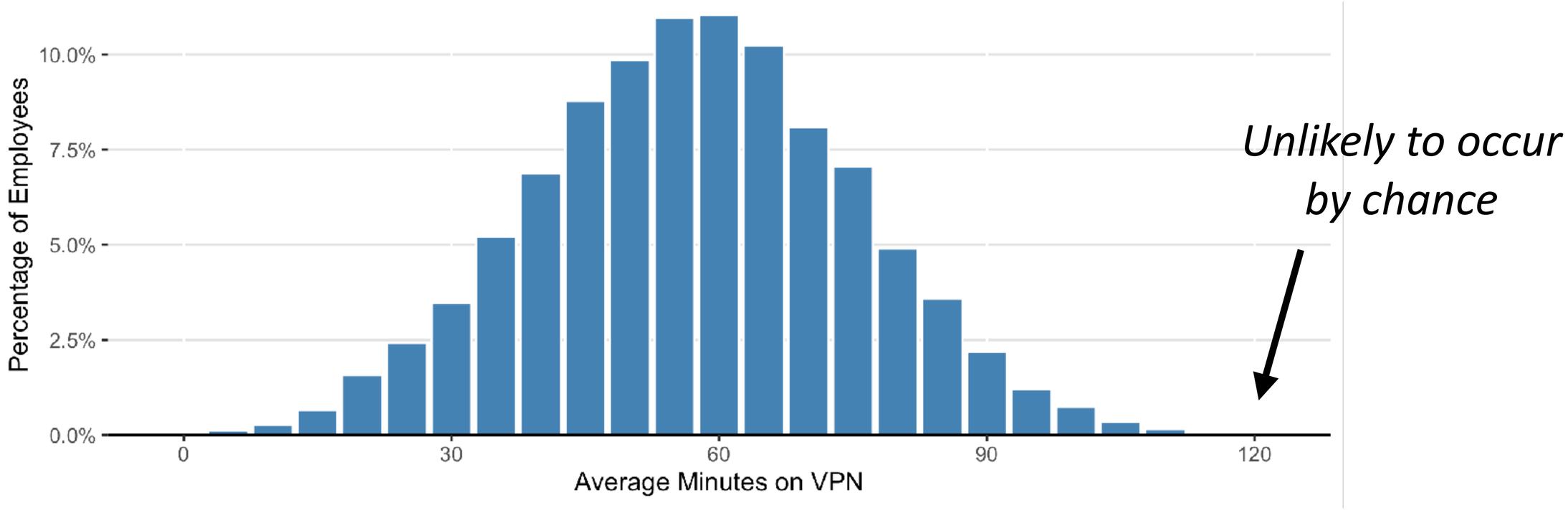


Hypothesis Testing: p-value

P-Value:

The probability of obtaining a result equal to or "more extreme" than what was actually observed, assuming everything is “good”.

At some point, we must reject the assumption that things are “good”.

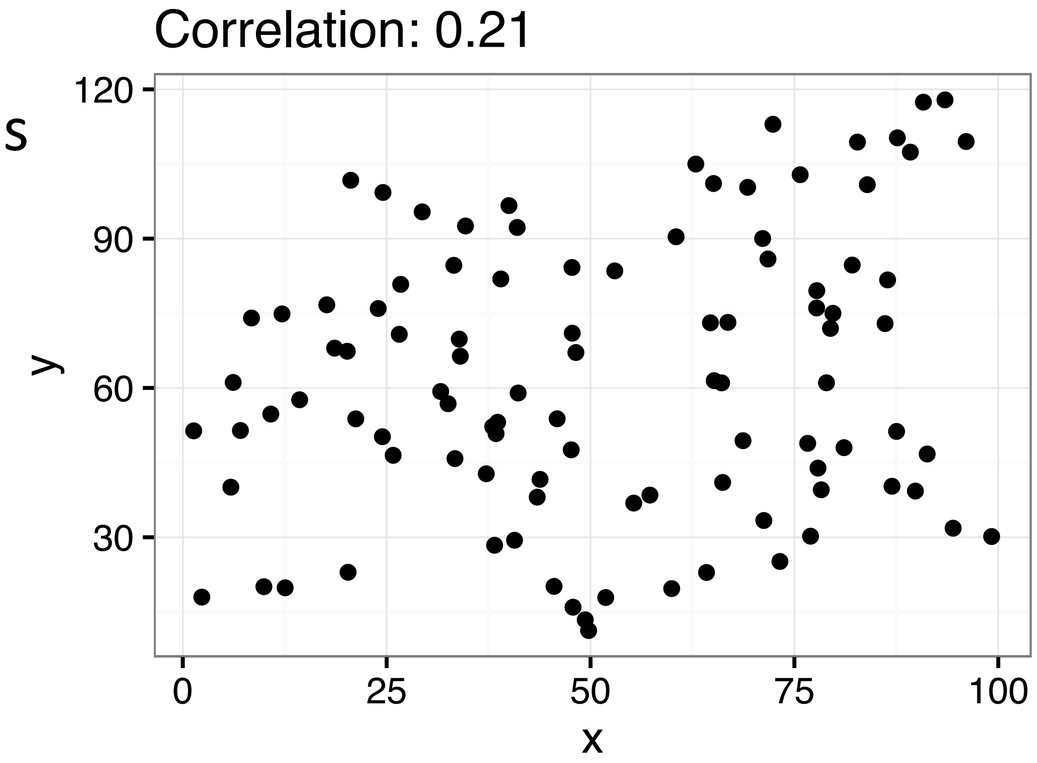


Hypothesis Testing: p-value

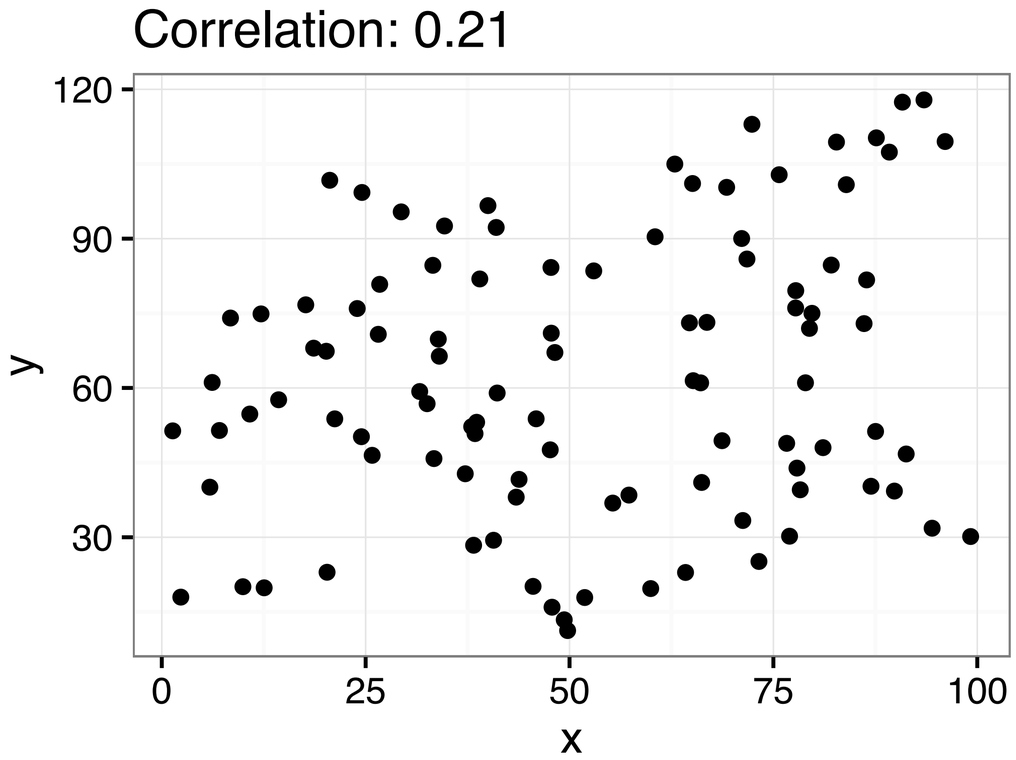
- With 100 values of each x and y, we estimate a correlation of 0.21
- If we assume the correlation is actually zero, what is the probability we would observe this (or more extreme) correlation?

p-value = 0.03671

- At “classic” threshold of 0.05 we would reject the “null hypothesis” (that the correlation is zero)
- “Big Data” messes with p-value because we can detect smaller and smaller differences.
- **P-Values are much more of a guide than a rule** (unless you are publishing)

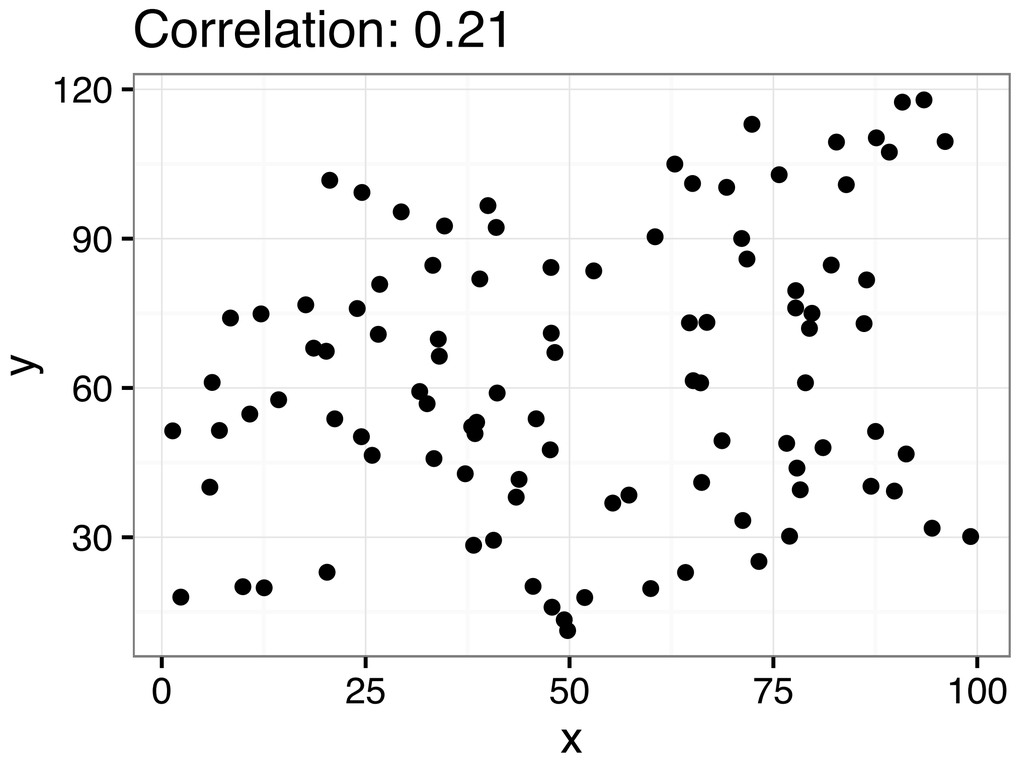


Correlation p-value in R




```
> cor.test(x,y)

   Pearson's product-moment correlation

data:  x and y
t = 2.118, df = 98, p-value = 0.03671
alternative hypothesis: true correlation is not equal to 0
… (more output coming up)
```

Correlation p-value in python




```
>series1.corr( series2 )
```

```
You can also specify kind='pearson'|'kendall'|'spearman'
```

Confidence Intervals

Last year, 21 of the 88 (23.8%) confirmed incidents were “severe”.
This year, 30 of 102 (29.4%) are severe.

Is there relative increase in the rate of “severe” incidents?

How sure are we of the answer?

	Severe	**Incidents**	**%**
Last year	21	88	23.8%
This year	30	102	29.4%

Confidence Intervals

Last year, 21 of the 88 (23.8%) confirmed incidents were “severe”.
This year, 30 of 102 (29.4%) are severe.

Is there relative increase in the rate of “severe” incidents?

How sure are we of the answer?

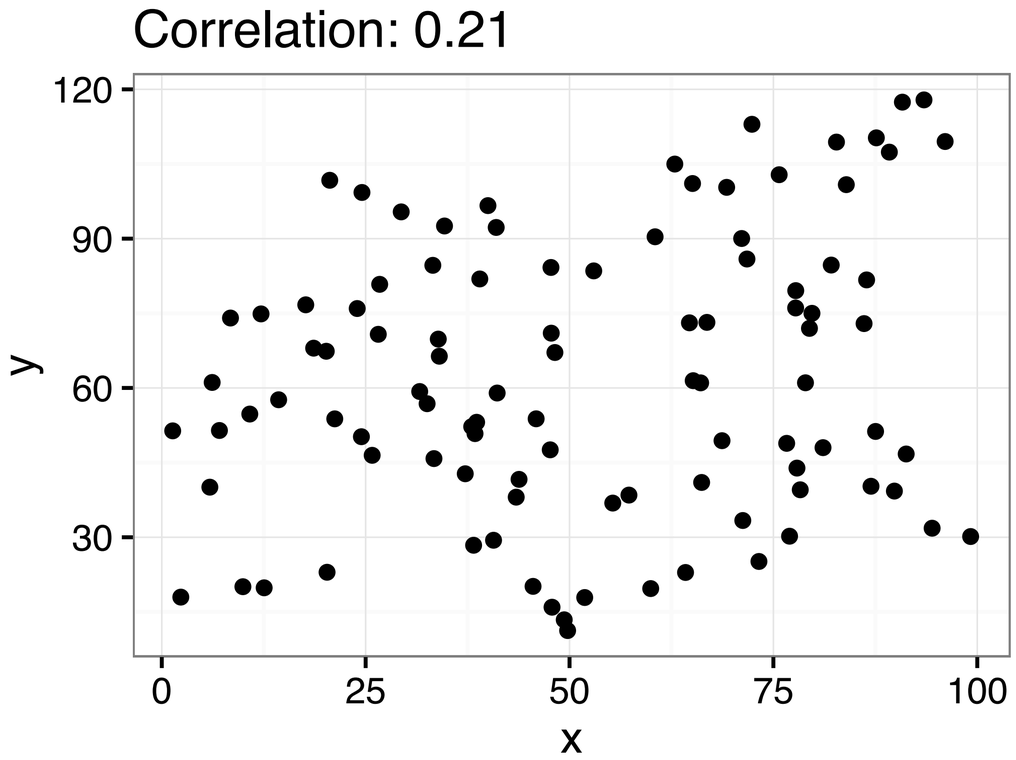
	Severe	**Incidents**	**%**	**95% lower**	**95% upper**
Last year	21	88	23.8%	15.4%	34.1%
This year	30	102	29.4%	20.8%	39.2%

In 95% of our samples, the “true” proportion of severe incidents will be between the lower and upper

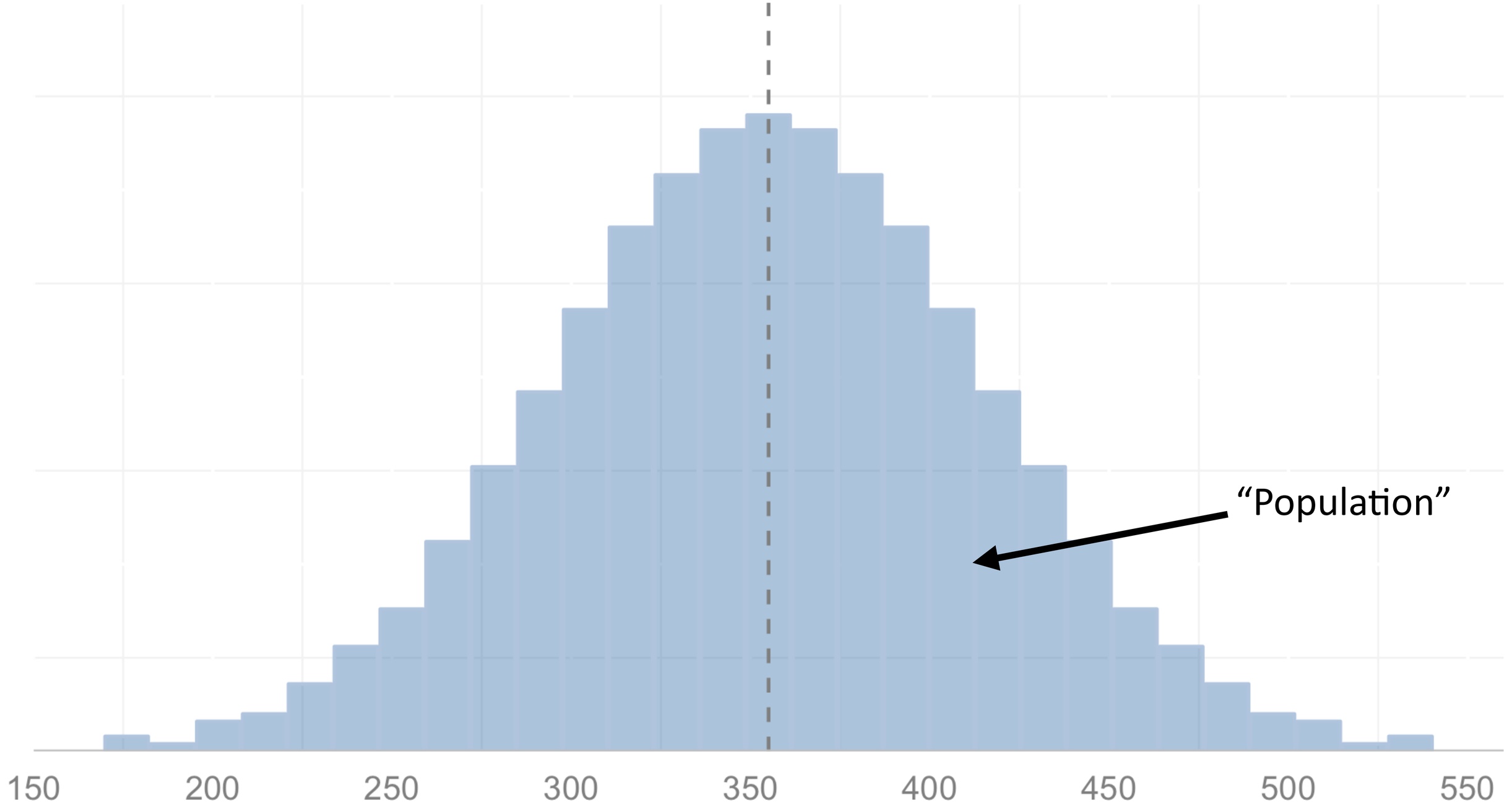
Correlation: Confidence Intervals

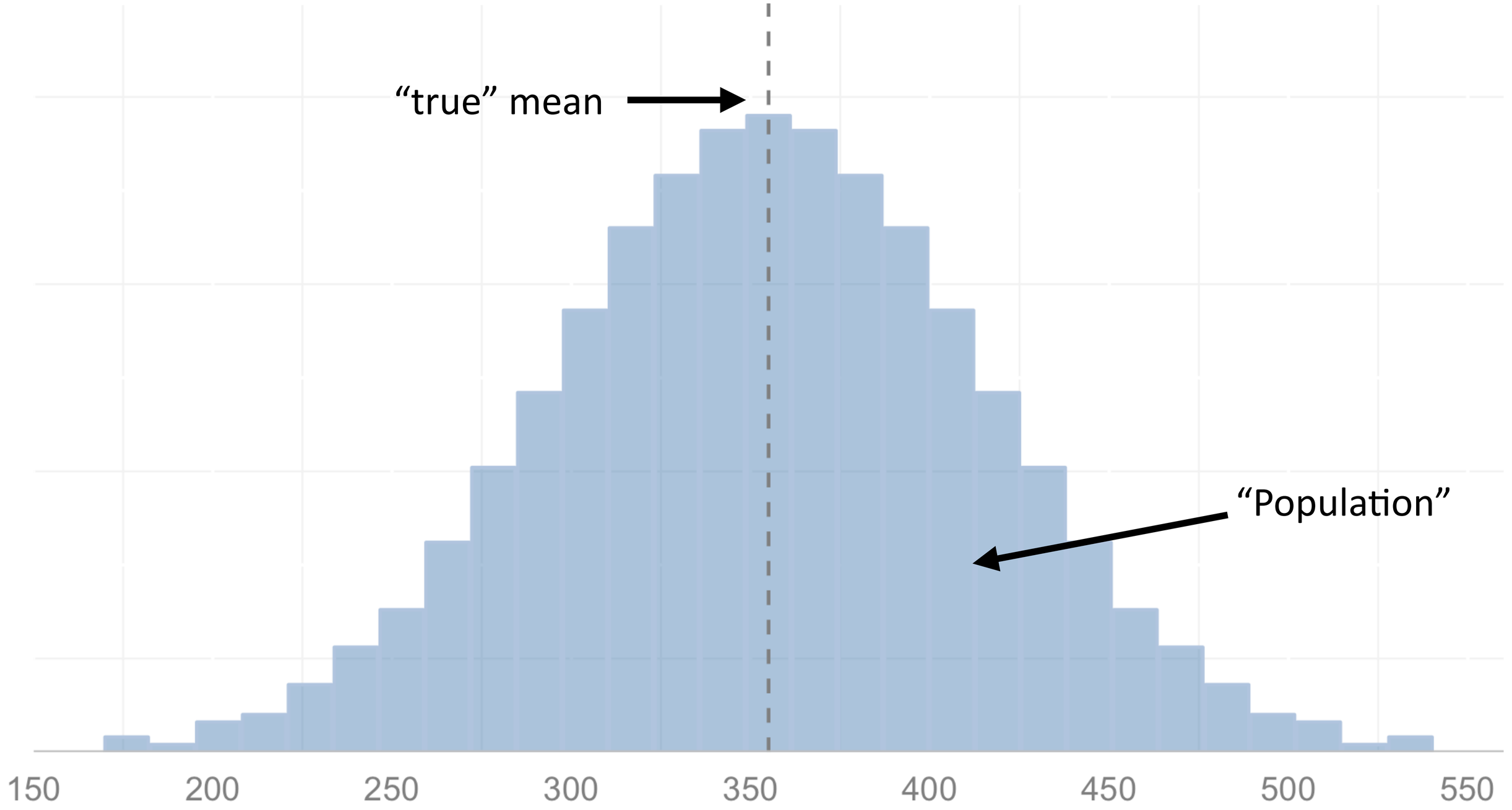
- With 100 values of each x and y, we estimate a correlation of 0.21
- What is the 95% confidence around "true" correlation?

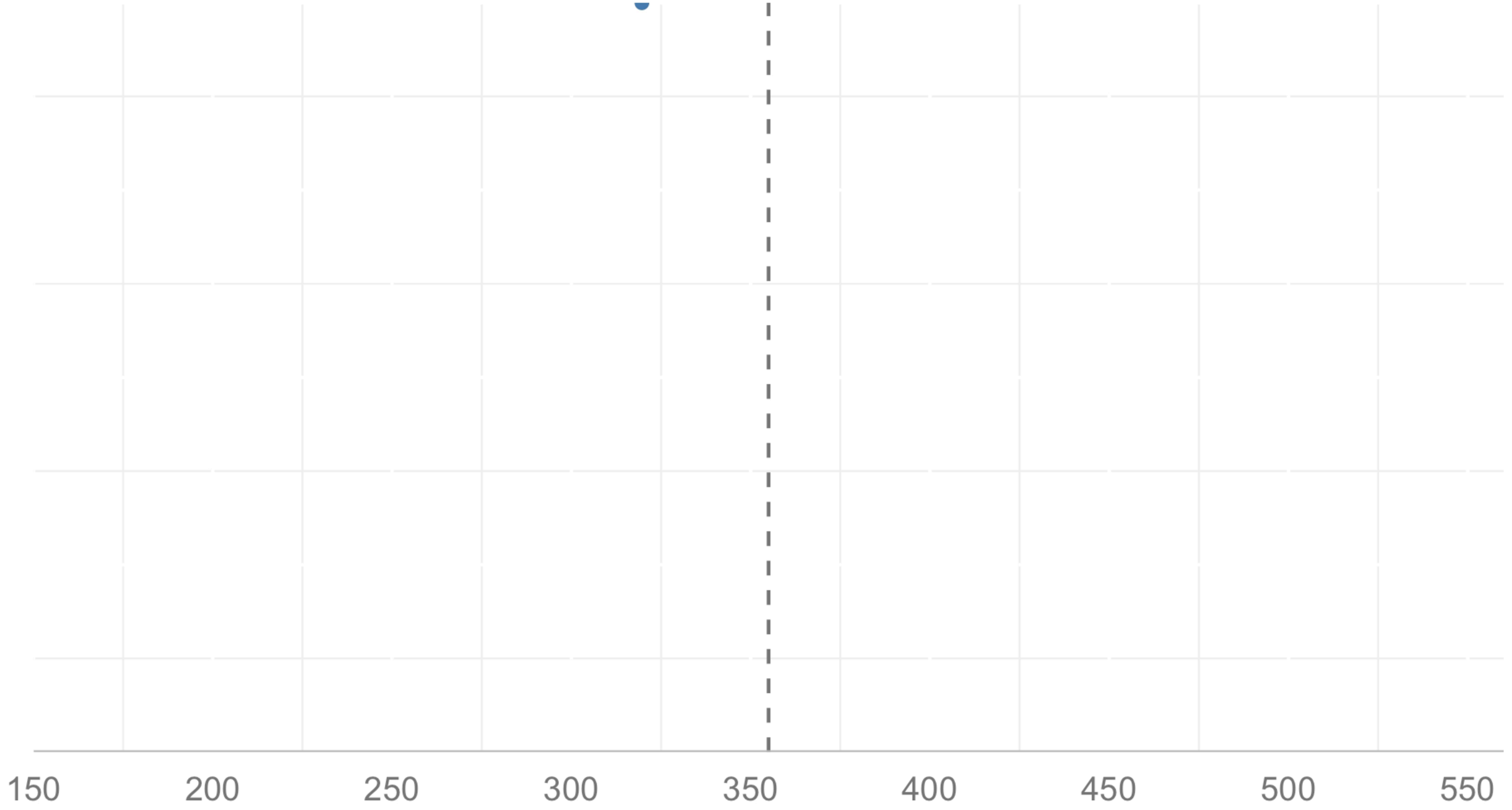
r is between 0.01 and 0.39

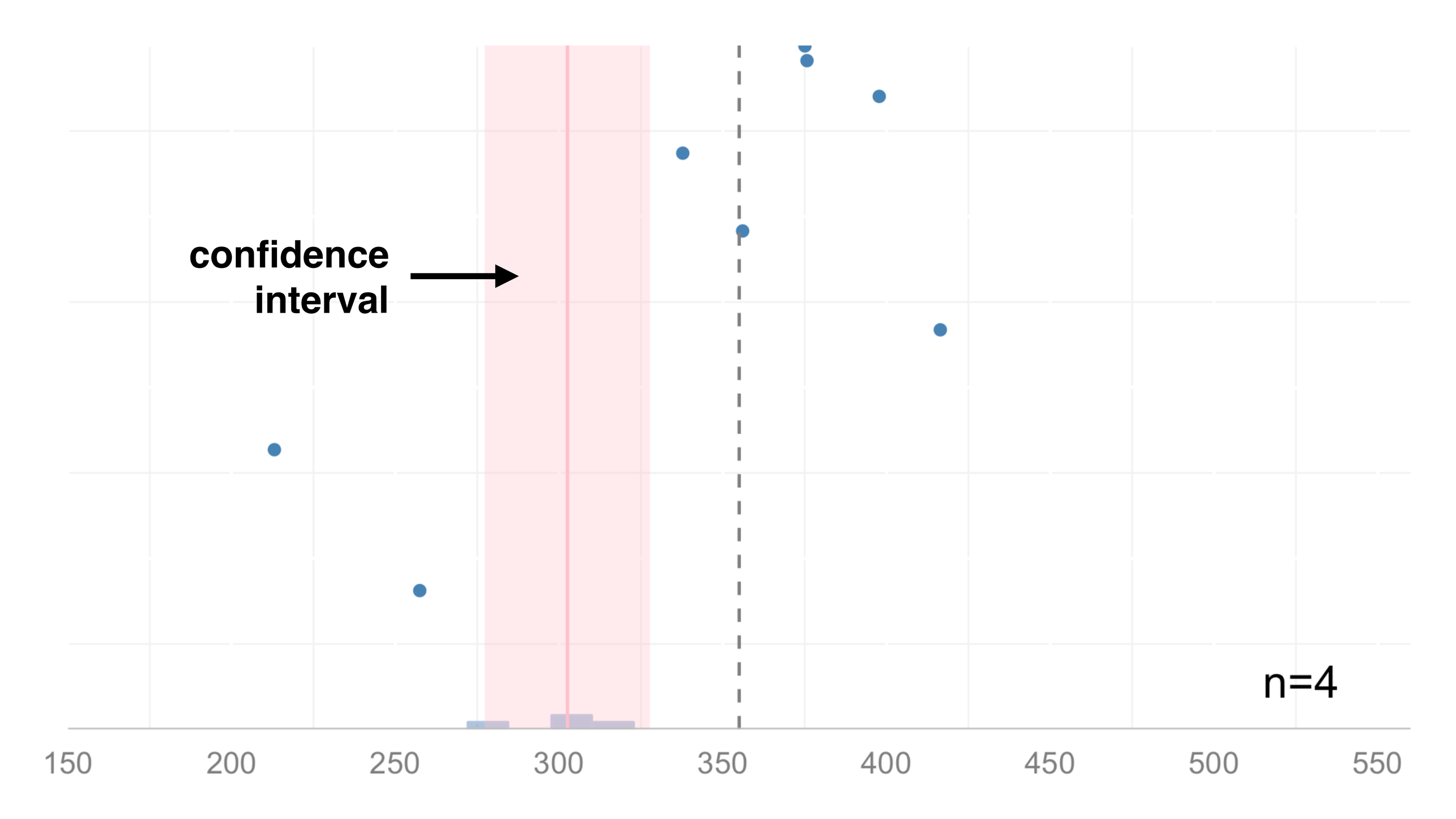


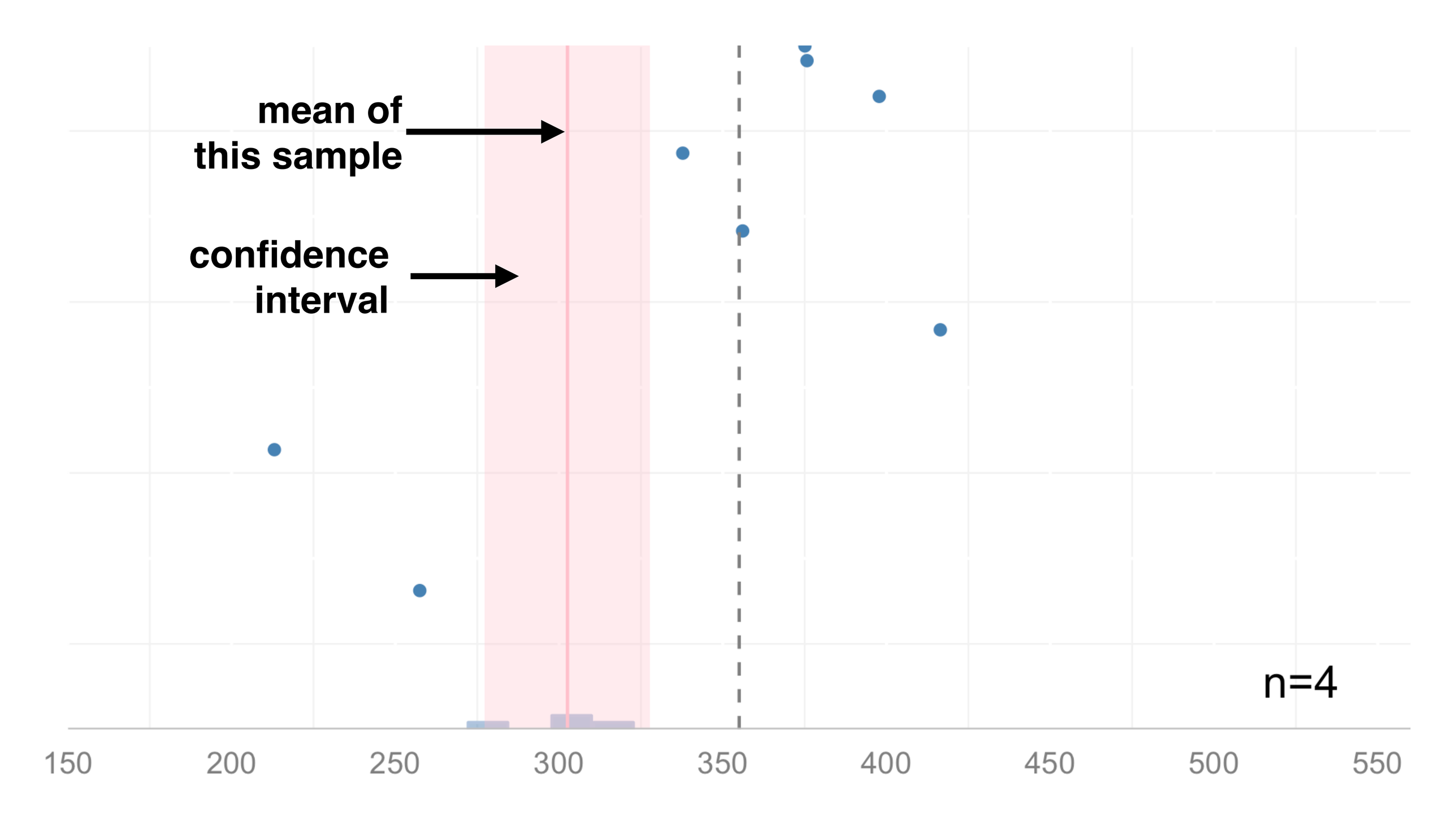

```
> cor.test(x,y)
… (hypothesis testing previously shown)

95 percent confidence interval:
 0.01334506 0.38962194
```

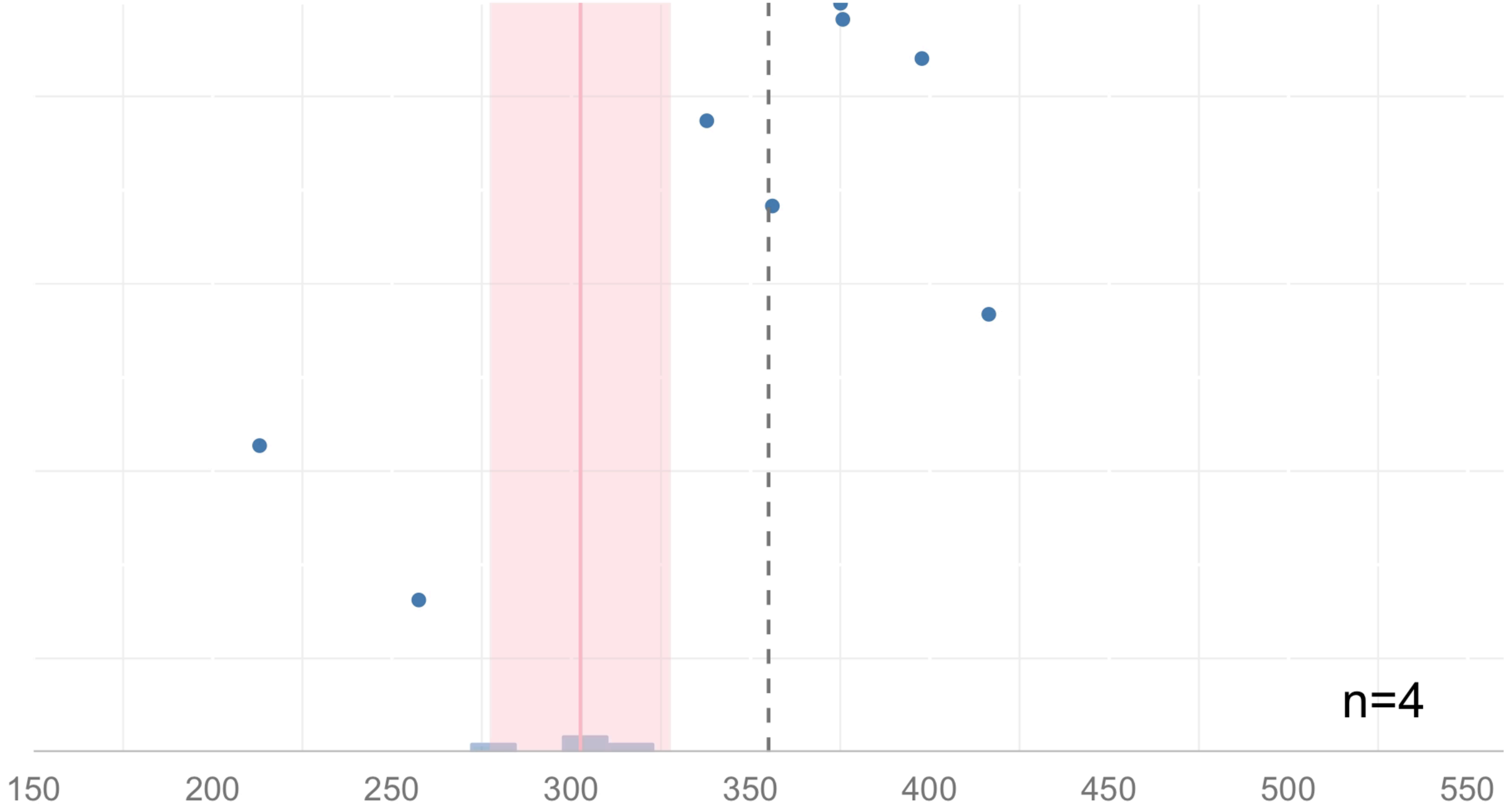

“Population”
150
200
250
300
350
400
450
500
550

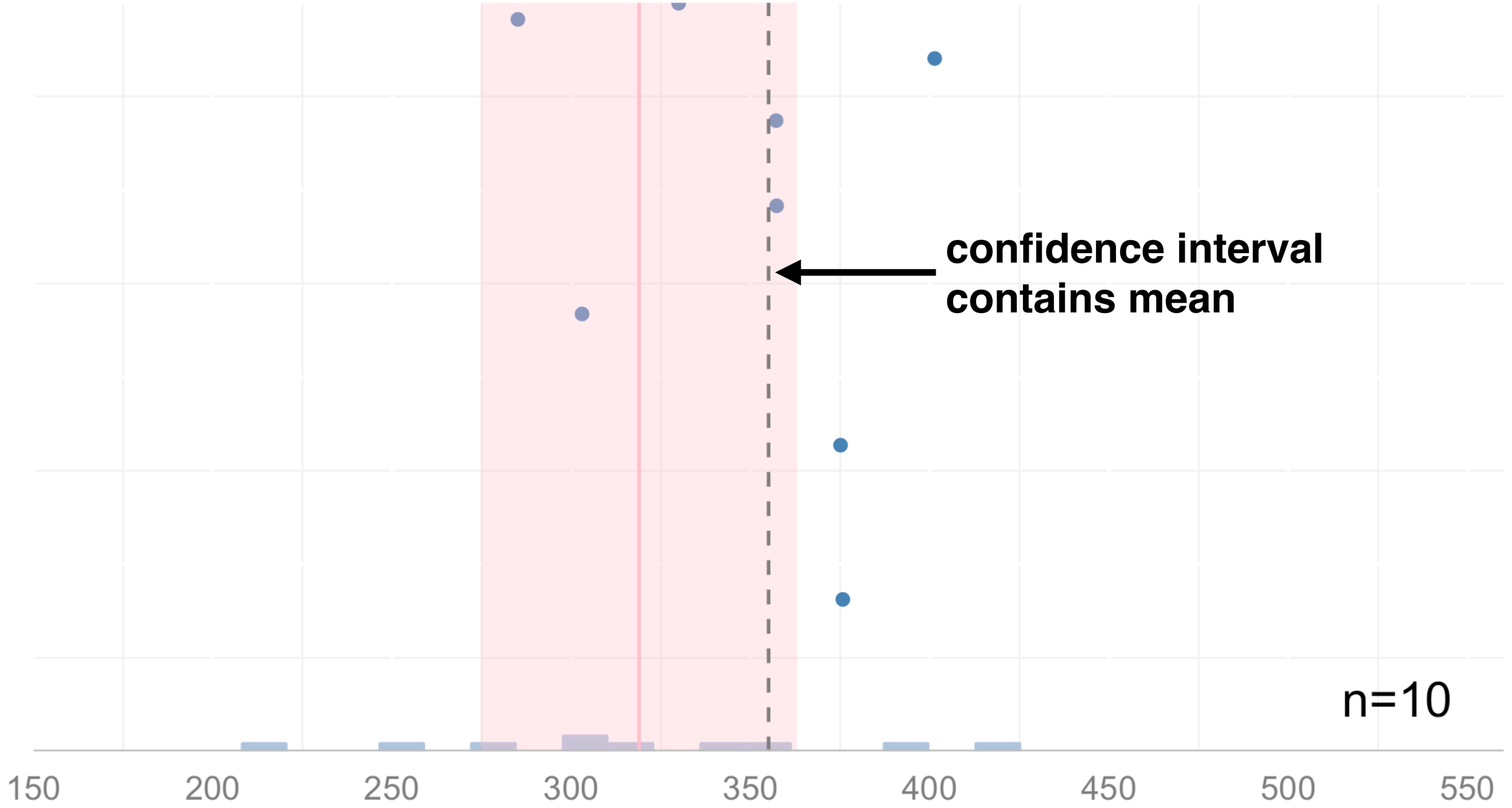

“true” mean
“Population”
150
200
250
300
350
400
450
500
550

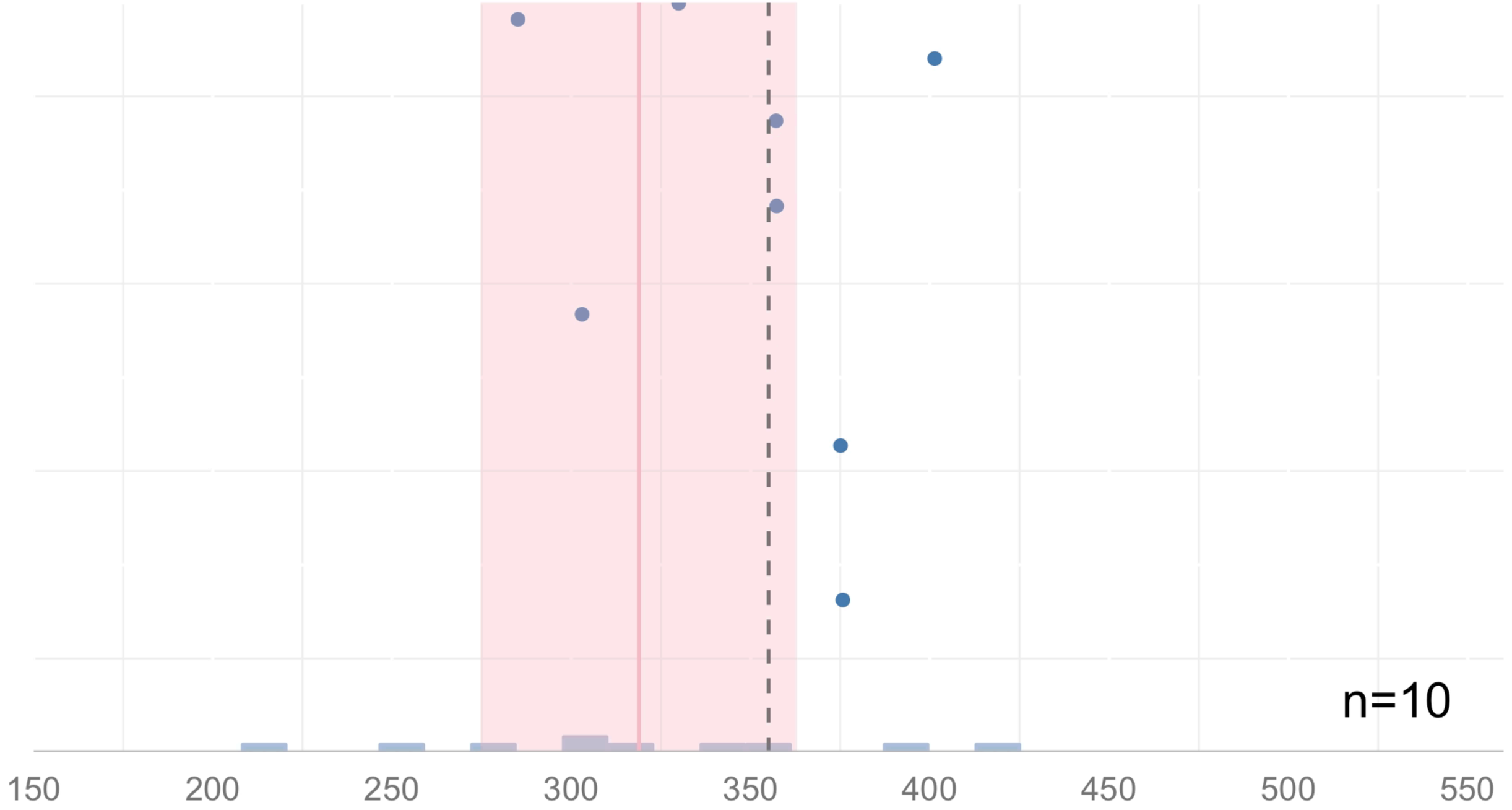

150
200
250
300
350
400
450
500
550

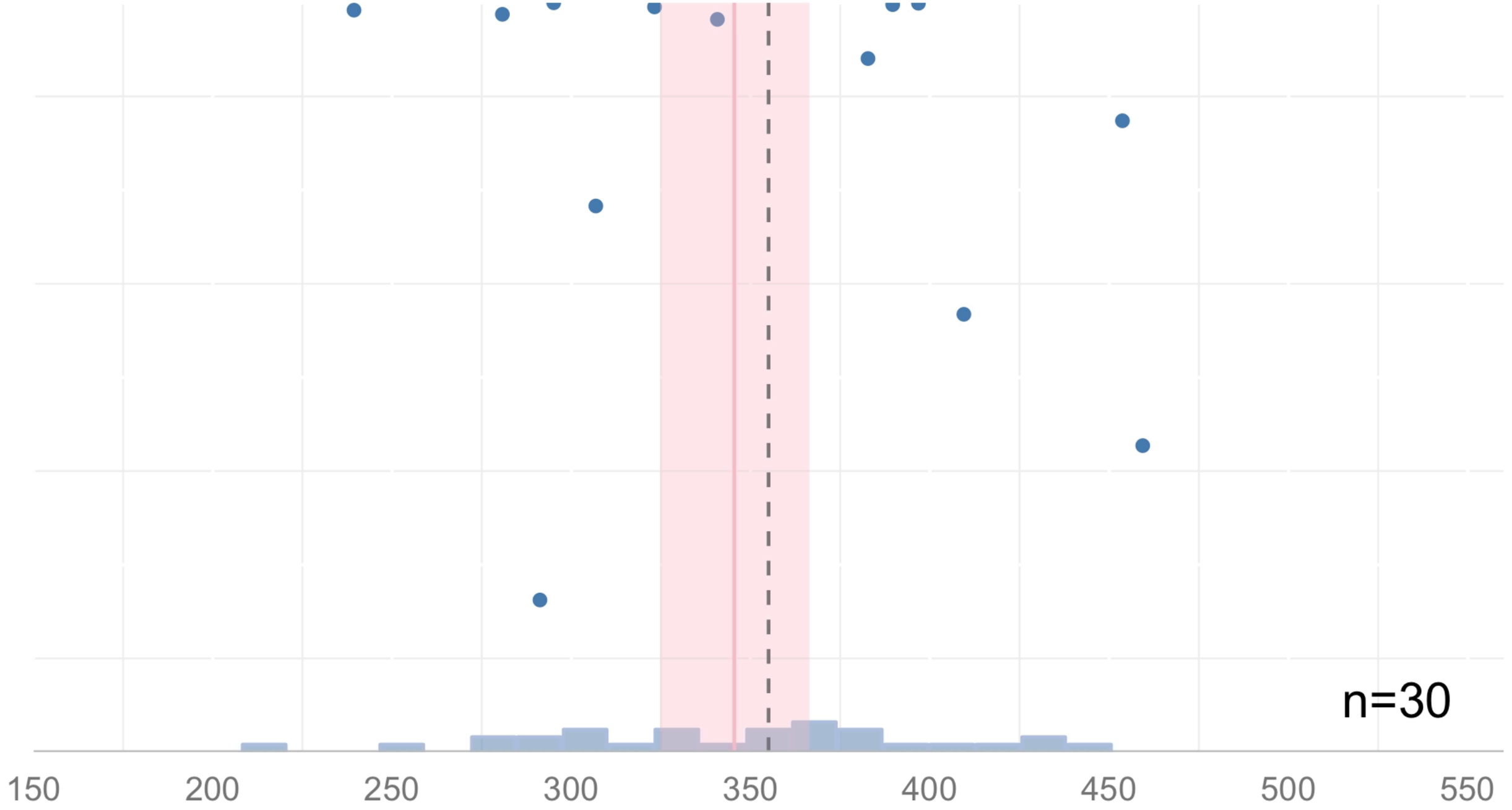

confidence
interval
n=4
150
200
250
300
350
400
450
500
550

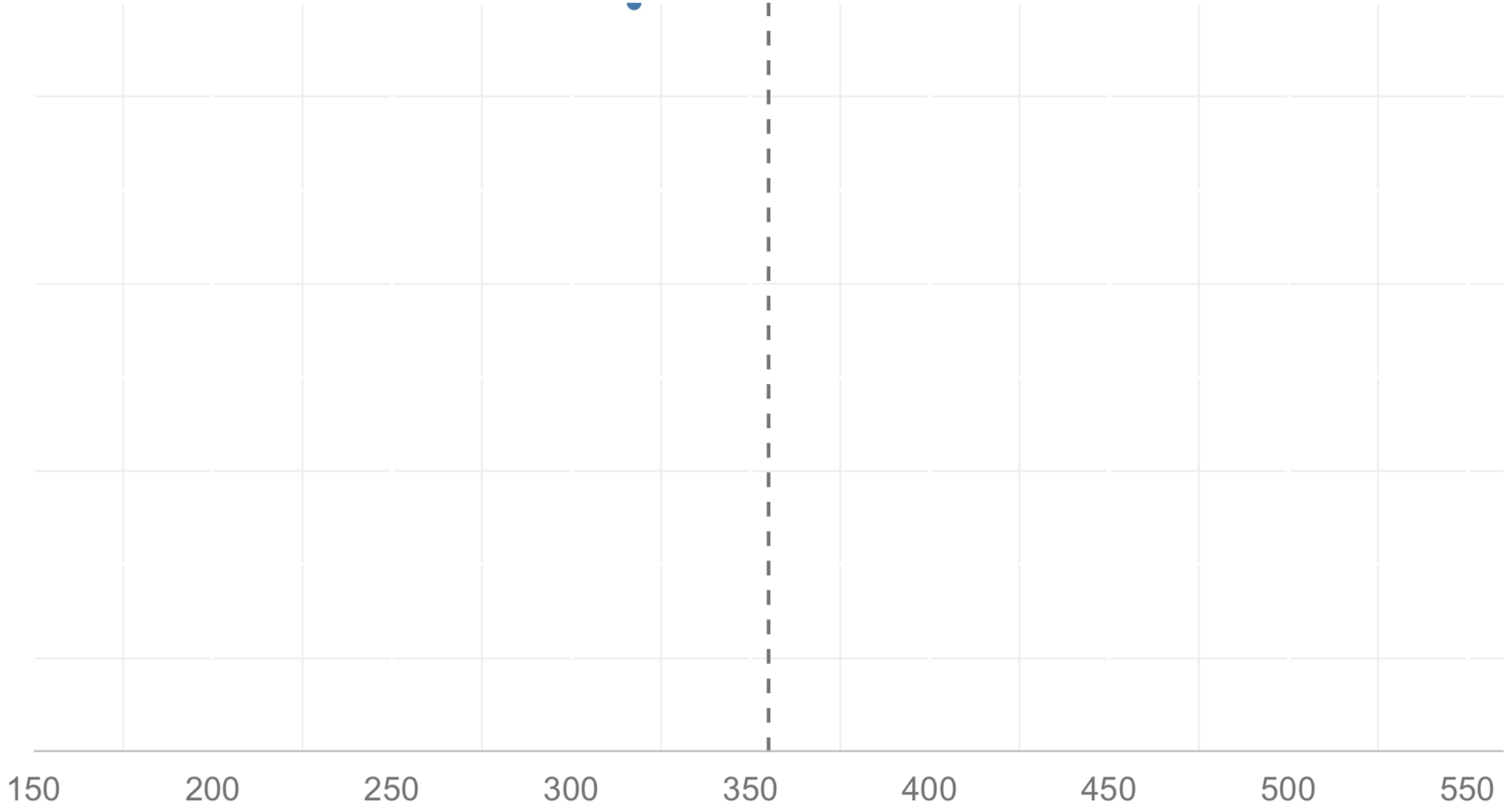

mean of
this sample
confidence
interval
n=4
150
200
250
300
350
400
450
500
550

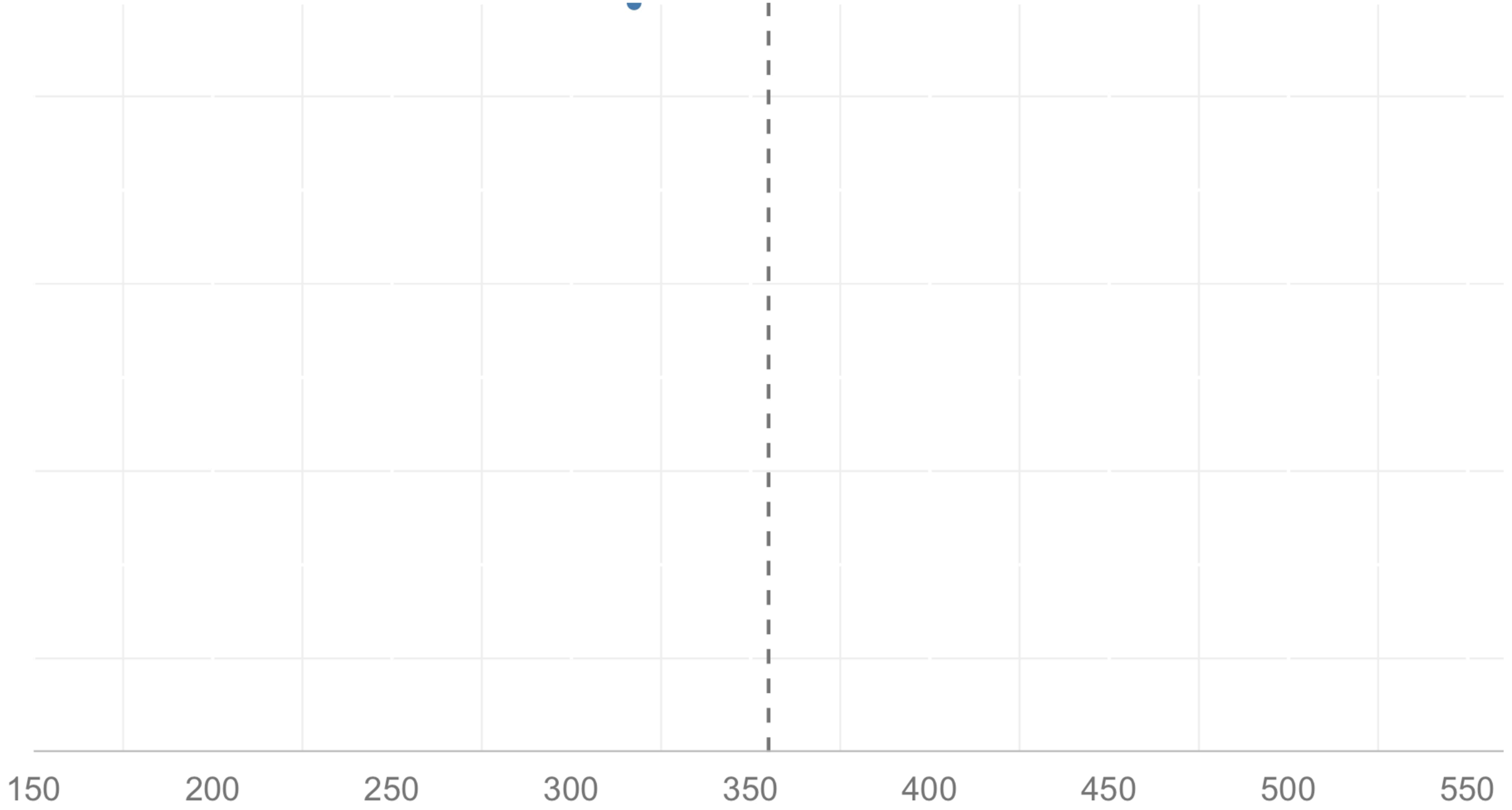

n=4
150
200
250
300
350
400
450
500
550

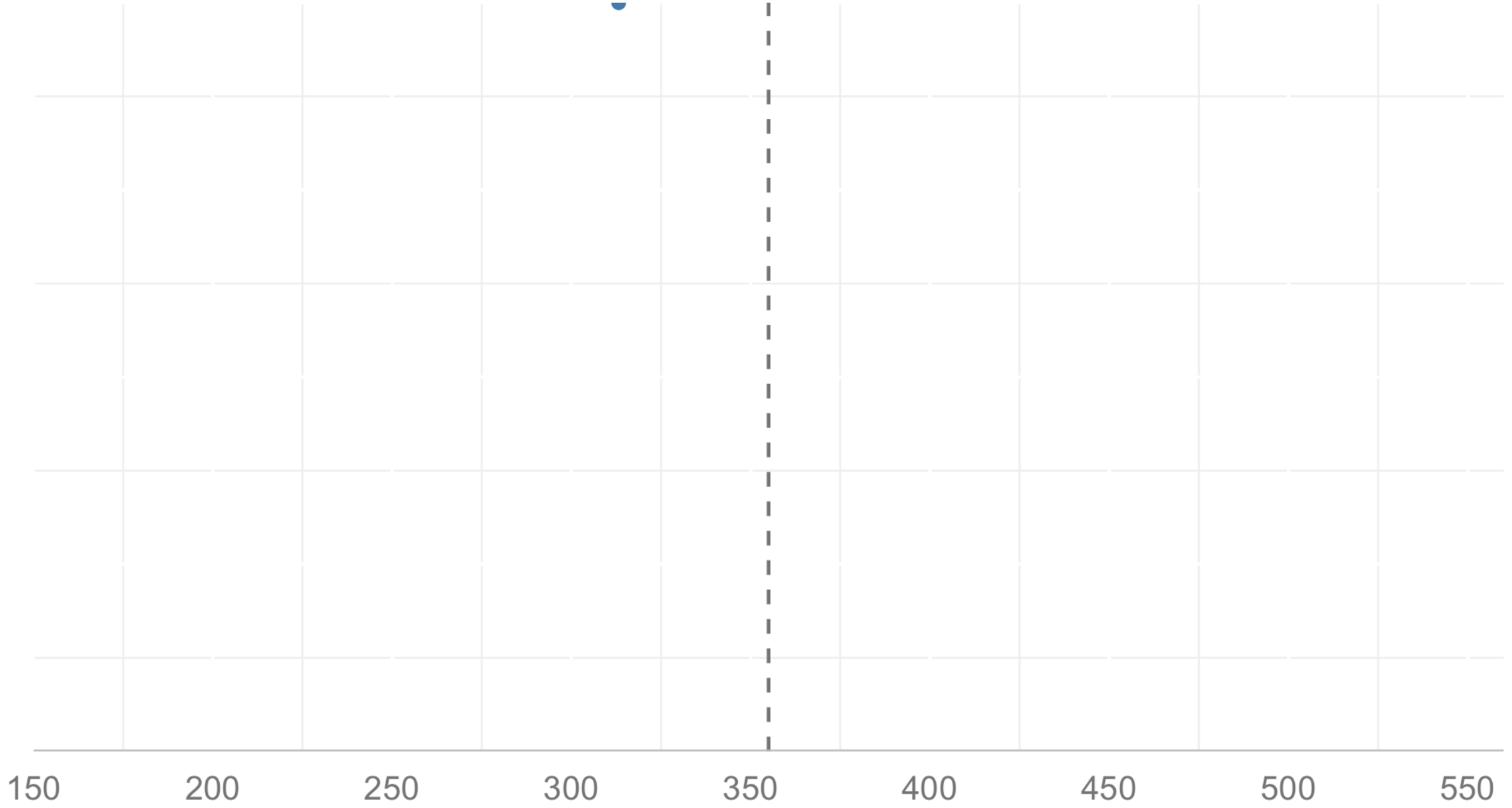

confidence interval
contains mean
n=10
150
200
250
300
350
400
450
500
550


n=10
150
200
250
300
350
400
450
500
550


n=30
150
200
250
300
350
400
450
500
550


150
200
250
300
350
400
450
500
550


150
200
250
300
350
400
450
500
550


150
200
250
300
350
400
450
500
550

O'REILLY®

Security

BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

EDA in code

Setting up for visualization

Python

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
pd.options.display.mpl_style = 'default'
```

R

```
library(tidyverse)  # will load ggplot2

# OR call them directly
library(ggplot2)
```

R: dplyr (tidyverse)

- "Code the way you think"

```
iris %>%
 group_by(Species) %>%
 summarise(avg = mean(Sepal.Width)) %>%
 arrange(avg)
```

- %>% basically puts the output on the left as the first argument on the right.

`x %>% f(y)` is the same as `f(x, y)`

`y %>% f(x, ., z)` is the same as `f(x, y, z)`

- Handful of core concepts:
  - **`filter`**: subset rows
  - **`select`**: subset columns
  - **`group_by`**: set variable(s) as group
  - **`summarize`**: simplify by grouping
  - **`mutate`**: modify/add variable
  - **`arrange`**: sort by variable(s)

https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf
Search for "dplyr cheat sheet"

Setting up for visualization

Python

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
pd.options.display.mpl_style = 'default'
```

R

```
library(tidyverse)  # will load ggplot2

# OR
library(ggplot2)
```

matplotlib

I know we have slides somewhere on matplotlib, I think in the data viz

ggplot2

- ggplot is based on a “grammar of graphics” from Wilkinson.
- ggplot is additive

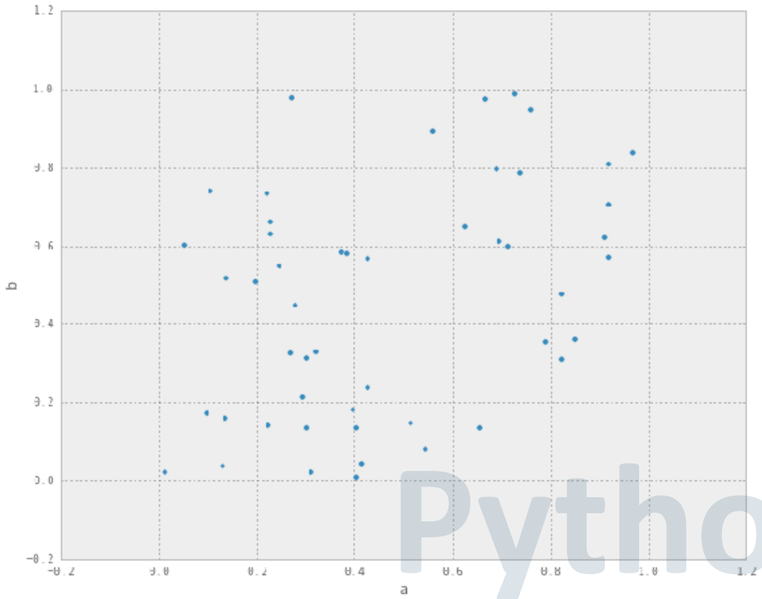
```
ggplot(<dataframe>, aes(<mappings>)) +
  geom_points() +
  scale_x_continuous(limits=c(0,10)) +
  theme_bw()
```

- ggplot supports simple plots, but is completely customizable
- ggplot can create Print-quality graphics

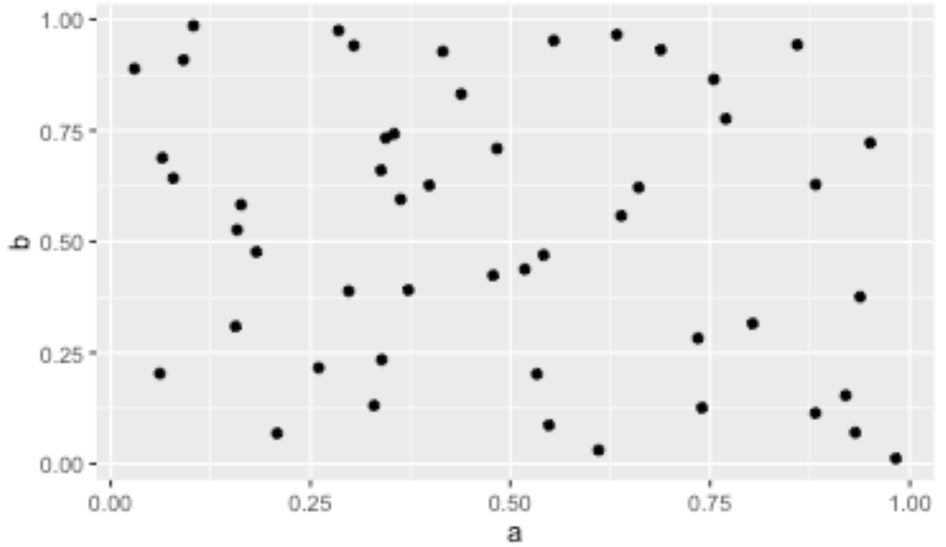
- **ggplot** - main function, sets up object
- **geoms** - “geometric objects”
  - geom_point, geom_bar, geom_histogram, geom_area, geom_line, geom_density, etc
- **aes** - aesthetic mappings
  - x, y, shape, color, fill, etc
- **scales** - Define rules of mapping
  - x, y, shape, color, fill, etc
- **theme** - customizes the visual presentation

Scatter Plot

```
df5 = pd.DataFrame(np.random.rand(50, 4),
columns=['a', 'b', 'c', 'd'] )
df5.plot(kind='scatter', x='a', y='b' )
```

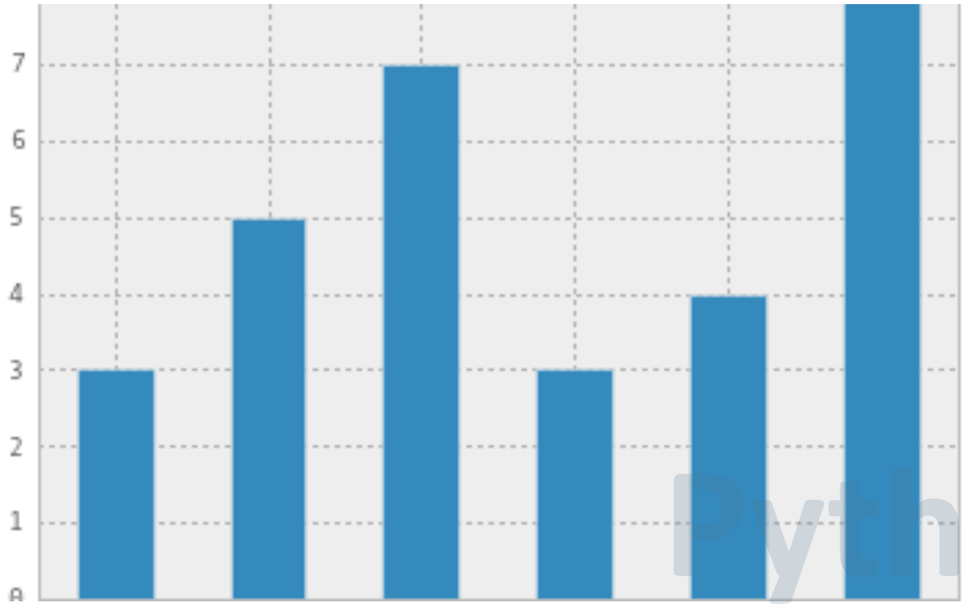



```
df5 <- data.frame(a=runif(50), b=runif(50),
                  c=runif(50), d=runif(50))
ggplot(df5, aes(x=a, y=b)) + geom_point()
```



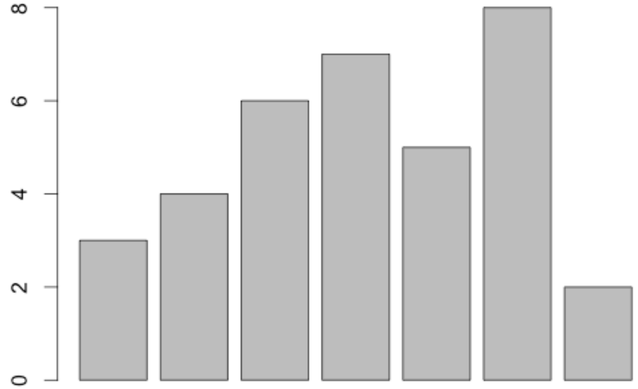
Setting up for visualization

```
barchart = data.plot( kind="bar" )
```



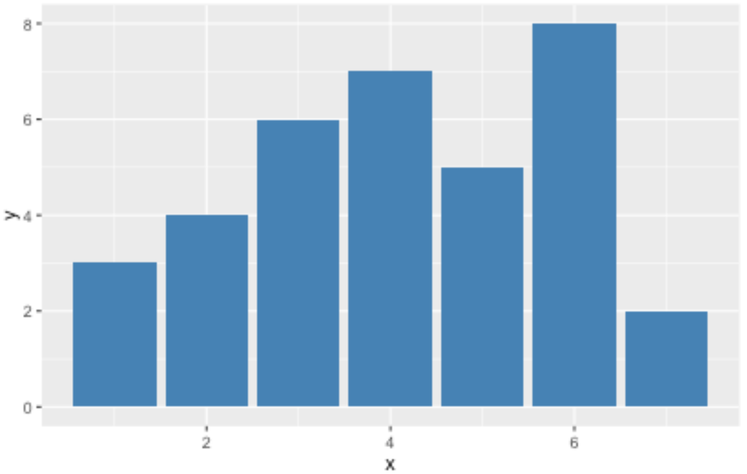
Python

```
barplot(data)
```



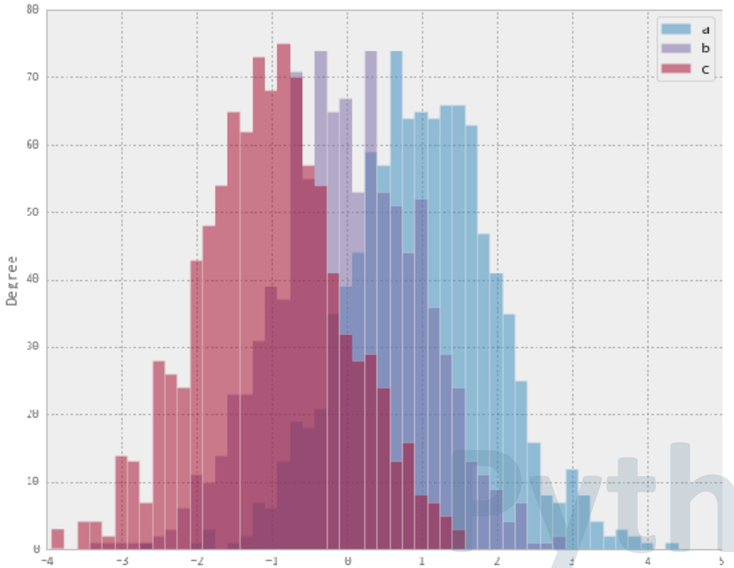
R

```
ggplot(dataf, aes(x, y)) +
geom_bar(stat="identity", fill="steelblue")
```

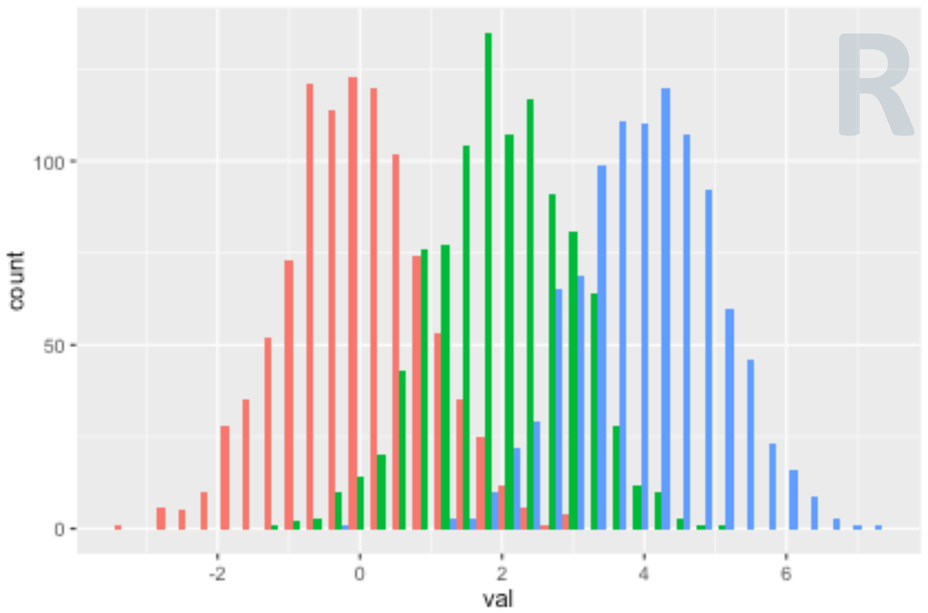


Histograms

```
df4.plot(kind='hist',
         alpha=0.5,
         bins=50 )
```




```
ggplot(df4, aes(val, fill=cat)) +
  geom_histogram(position="dodge",
                 binwidth = 0.3)
```



O'REILLY®
Security
BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

Data Sources for this Training

Daily Bots

- **date** *(%Y-%m-%d)*
- **botfam**: Botnet Family
- **industry**: Industry of detected infection
- **hosts**: Number of infected hosts for the day, bot family and industry
- **orgs**: Number of infected orgs for the day, bot family and industry

	industry	hosts
1	Manufacturing	256701
2	Retail	218970
3	Education	76455
4	Government/Politics	63701
5	Finance	57387
6	Healthcare/Wellness	33915

	botfam	orgs
1	Conficker AB	34039
2	Sality	8591
3	Necurs	8554
4	Ramnit	7940
5	Zeus	5394
6	Bedep	4289
7	zeroaccess	2949
8	Zusy	2629
9	PushDo	1205
10	Olmasco	492

Daily Bots

	date	botfam	industry	hosts	orgs
1	2016-06-01	Bedep	Education	88	33
2	2016-06-01	Bedep	Finance	387	17
3	2016-06-01	Bedep	Government/Politics	430	42
4	2016-06-01	Bedep	Healthcare/Wellness	42	19
5	2016-06-01	Bedep	Manufacturing	184	18
6	2016-06-01	Bedep	Retail	684	11
7	2016-06-01	ConfickerAB	Education	372	93
8	2016-06-01	ConfickerAB	Finance	198	56
9	2016-06-01	ConfickerAB	Government/Politics	255	38
10	2016-06-01	ConfickerAB	Healthcare/Wellness	224	109

... with 4,959 more rows

Raw DGA Data set

"alldomains.csv"

- host: domain + top level domain
- src: Botnet host belongs to (or "Alexa" for non-botnet)

	src	count
1	alexa	50000
2	bamital	10000
3	cryptolocker	10000
4	gameoverdga	10000
5	necurs	10000
6	nivdort	10000

host	src
newyorkpass.com	alexa
techcrunch.com	alexa
c92d464ec85397f9635955089ee30ff2.info	bamital
3b1976d97c9f07fc4e7b3070b3740fc4.co.cc	bamital
ynhjqpyepyvaibh.net	cryptolocker
foxhndkbdseb.org	cryptolocker
1vfglaw1ojizj415wzi4u1h527xy.org	gameoverdga
ja6aeb14q07sr1kb91c41w62ri9.org	gameoverdga
dkmgrevbxhigrktki.ki	necurs
kqynxlusmendiqblwrt.bz	necurs
wifenews.net	nivdort
hairnerve.net	nivdort

100,000 rows

“dgasample”

Small sample of algorithmically generated domains (and alexa domains) with four features generated for each domain

```
     host                      src          domain
Length:600          alexa        :100   Length:600
Class :character    bamital      :100   Class :character
Mode  :character    cryptolocker:100   Mode  :character
                    gameoverdga  :100
                    necurs       :100
                    nivdort      :100

      len             numbers           dicts           entropy
Min.   : 3.00   Min.   : 0.000   Min.   :0.0000   Min.   :0.9183
1st Qu.: 9.00   1st Qu.: 0.000   1st Qu.:0.5000   1st Qu.:2.9477
Median :14.00   Median : 0.000   Median :0.6923   Median :3.3747
Mean   :17.29   Mean   : 4.728   Mean   :0.6891   Mean   :3.2972
3rd Qu.:26.00   3rd Qu.: 9.000   3rd Qu.:1.0000   3rd Qu.:3.6837
Max.   :32.00   Max.   :26.000   Max.   :1.0000   Max.   :4.4839
```

- **host:** domain + tld
- **src**: source for host
- **domain:** just domain string
- **len**: number of characters in domain
- **numbers**: count of numbers in domain
- **dicts:** percentage of domain explainable by a dictionary
- **entropy**: complexity of character set in domain

{LAB} time

Please complete “EDA Worksheet”