

O'REILLY®

# Security

BUILD BETTER DEFENSES

[oreillysecuritycon.com](http://oreillysecuritycon.com)

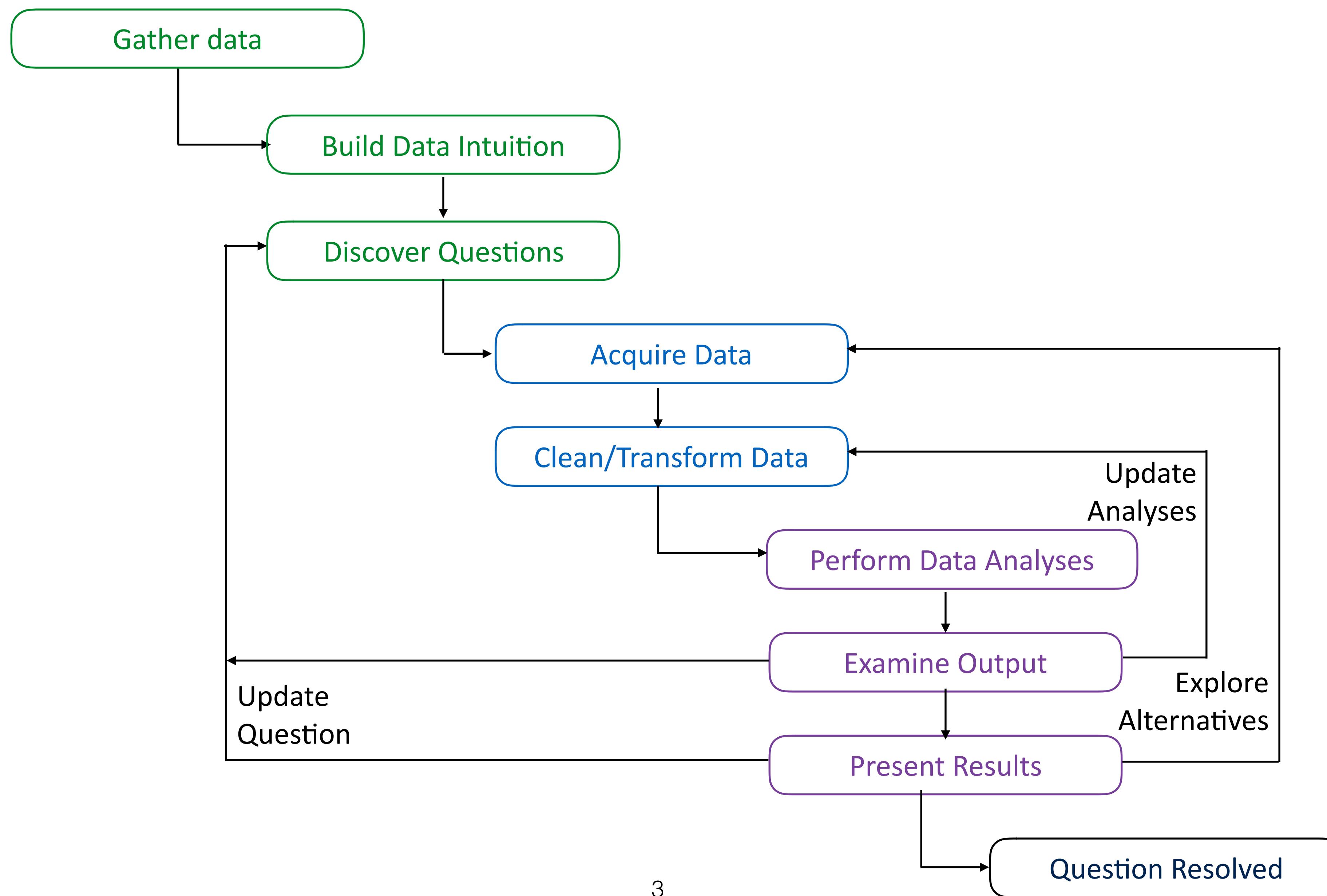
#oreillysecurity

# Foundations of Security Data Science Exploratory Data Analysis

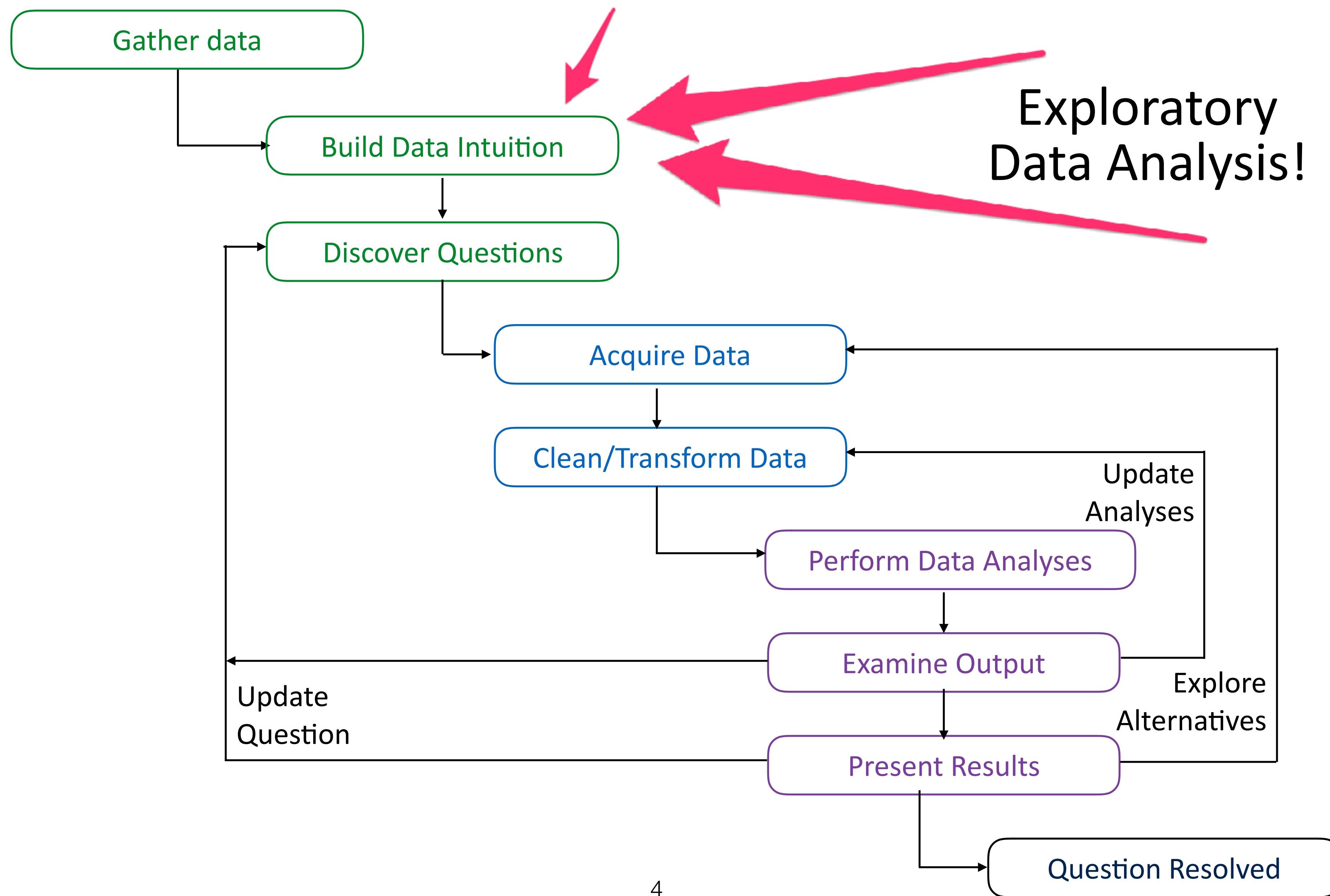
Jay Jacobs  
Charles Givre

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

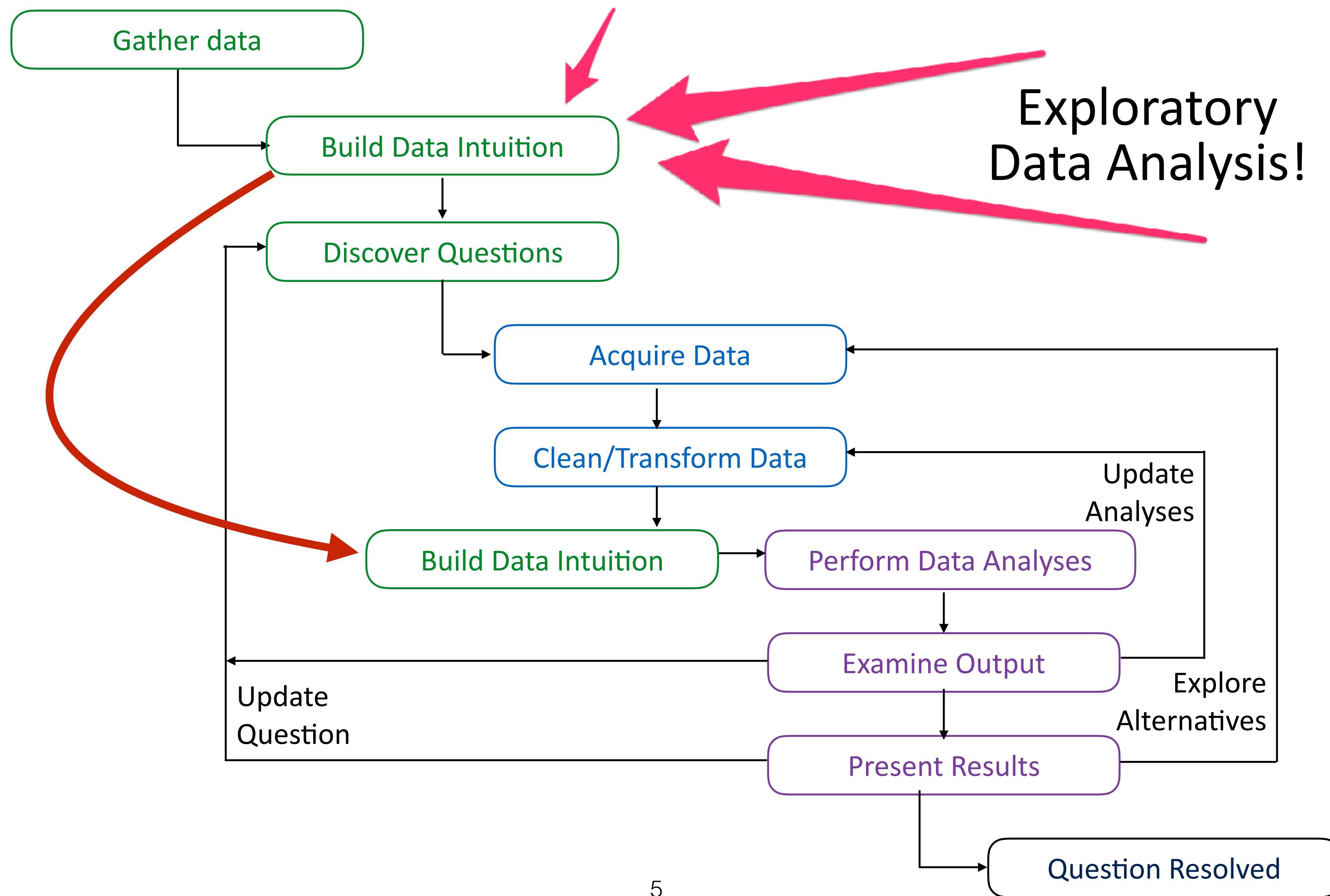
# Reality of Research Workflow



# 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

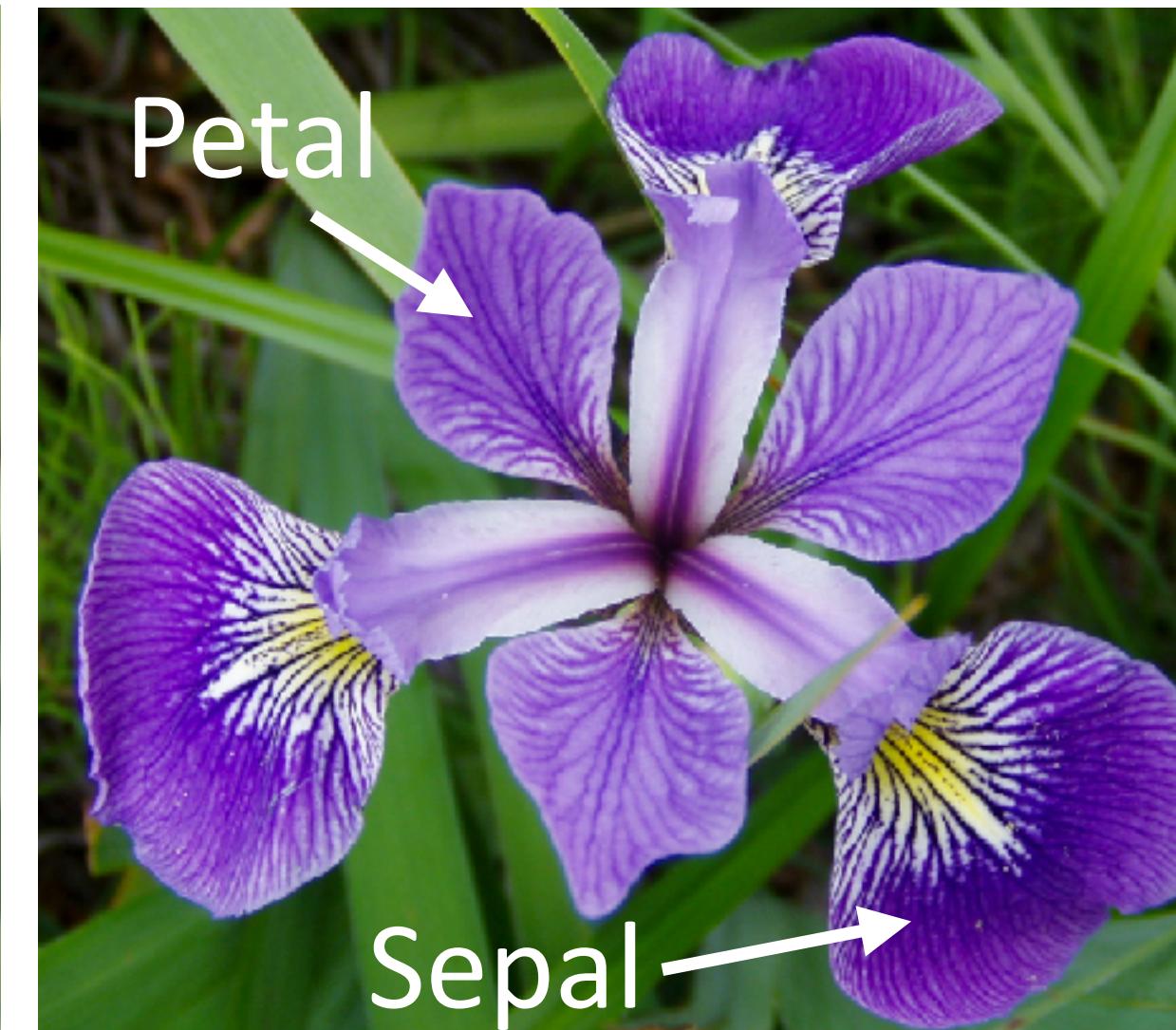
```
'data.frame': 150 obs. of 5 variables:  
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
 $ 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



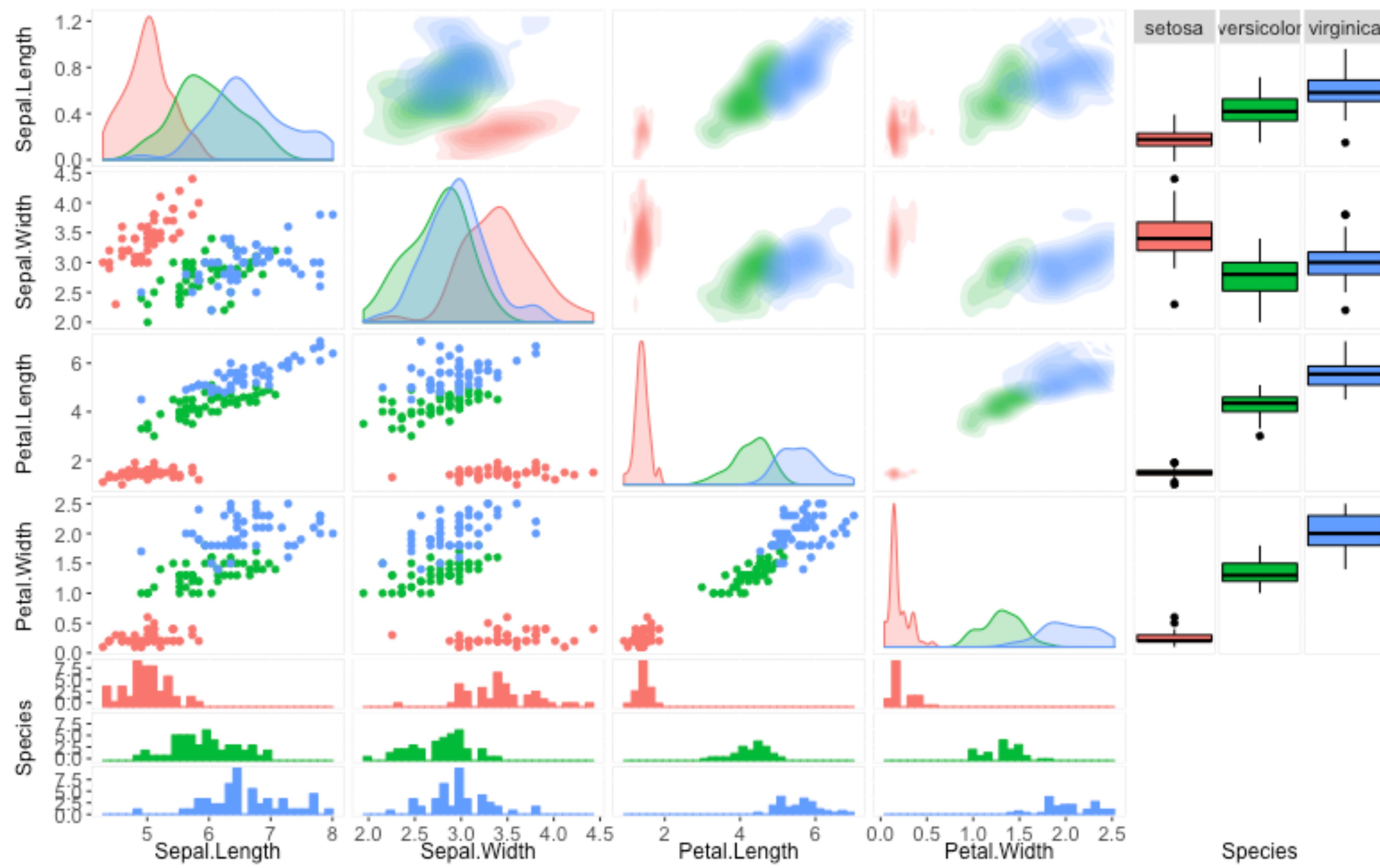
Iris Versicolor



Iris Virginica



# Iris



# Text Summaries

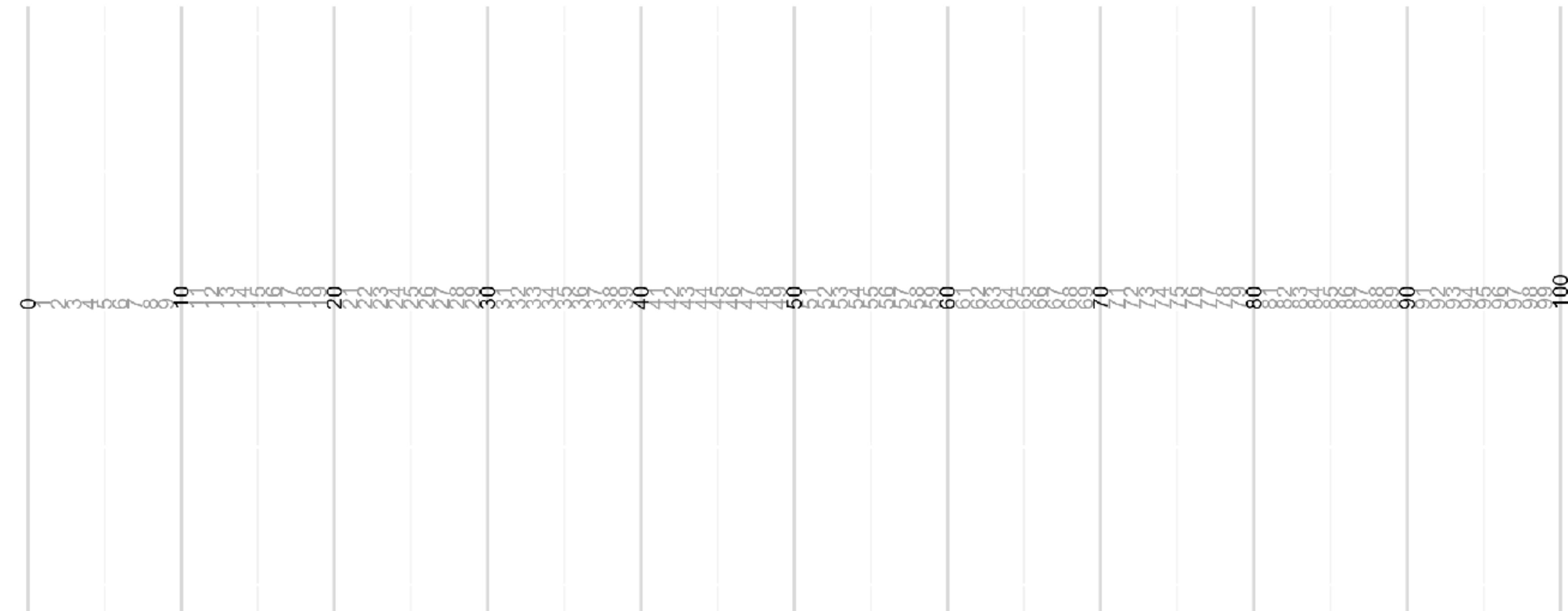
## Python (pandas) describe()

	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

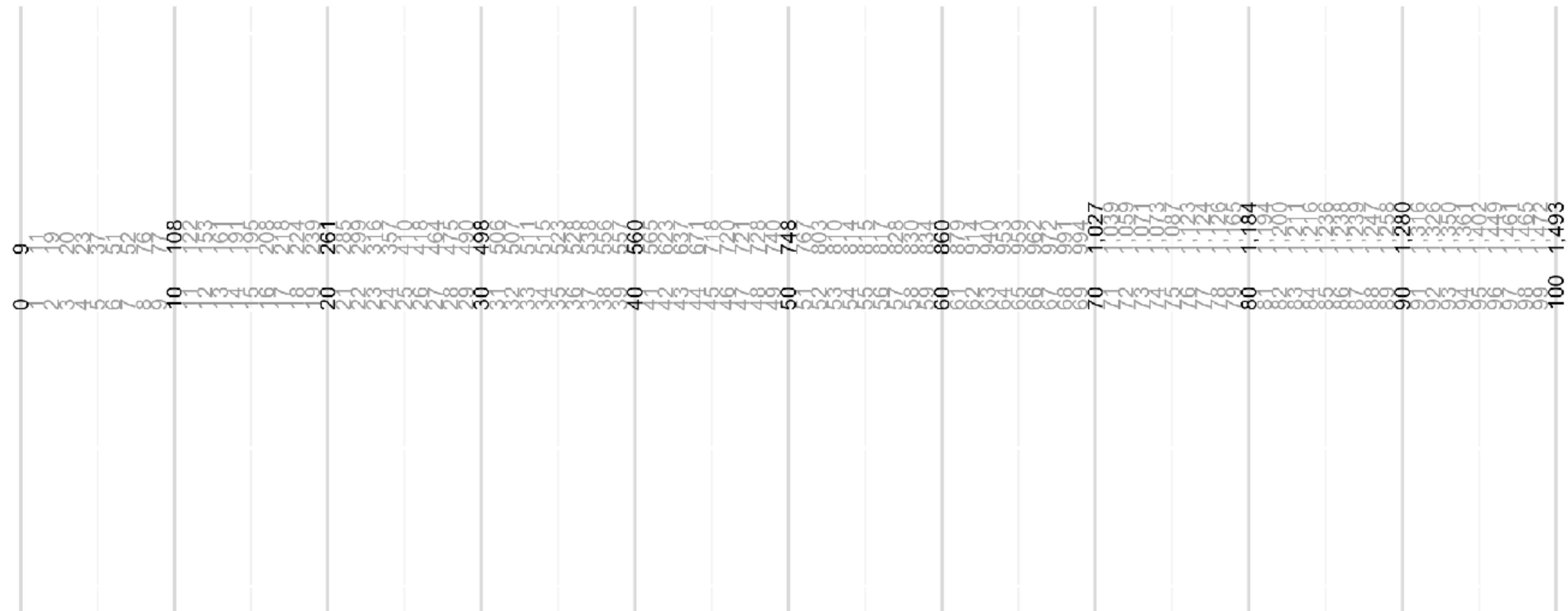
## R summary()

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	

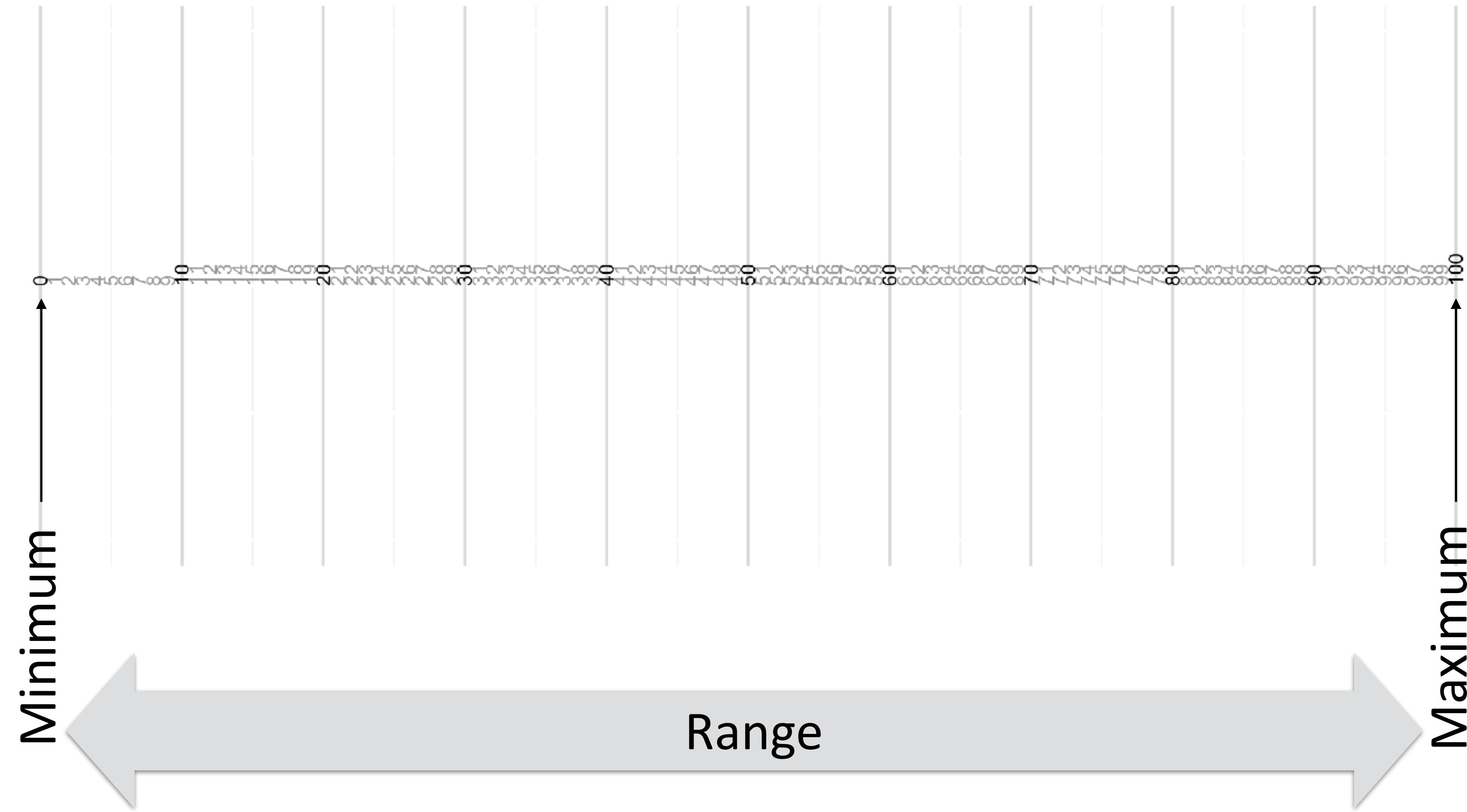
# Describing a Single Variable



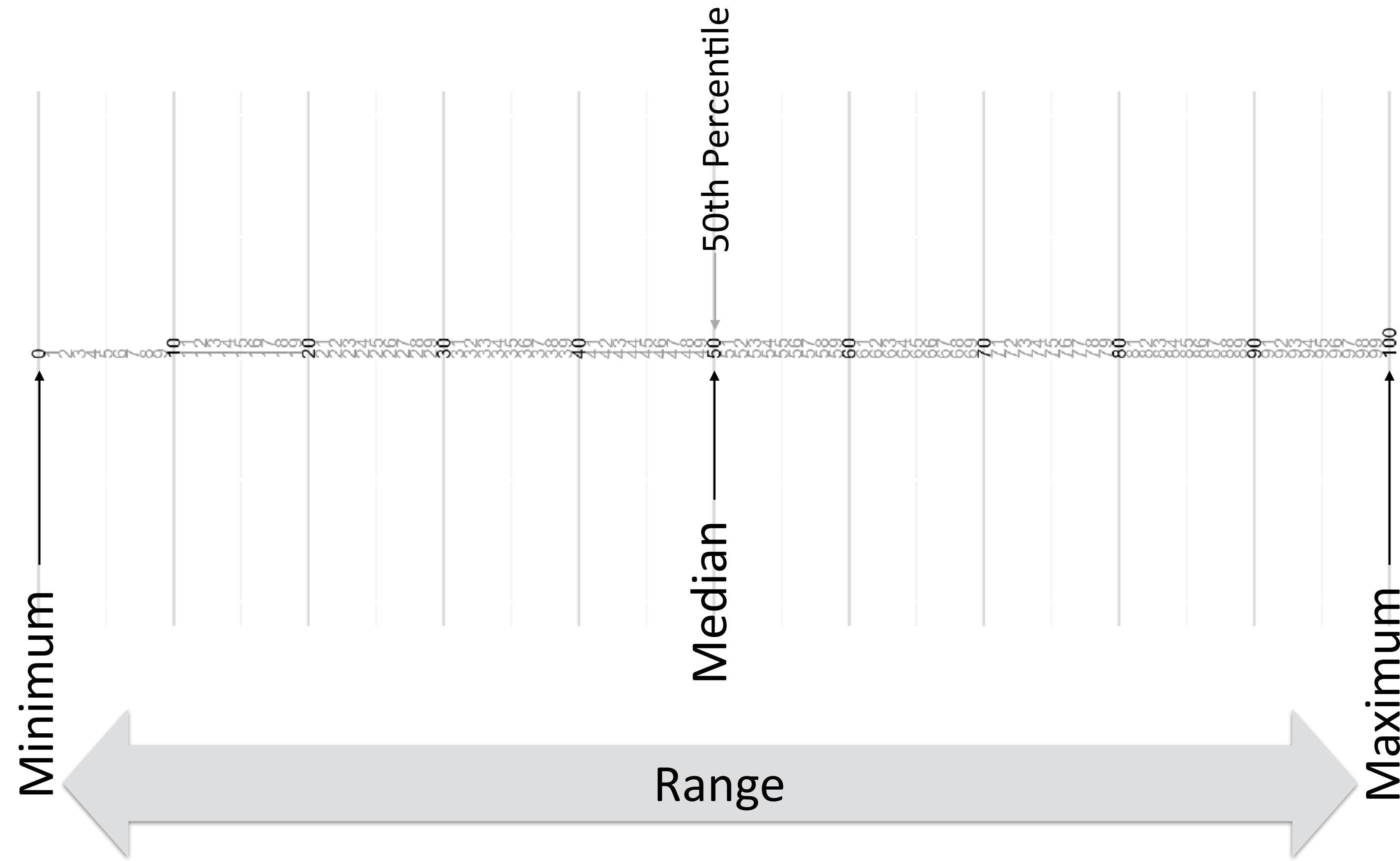
# Single Variable



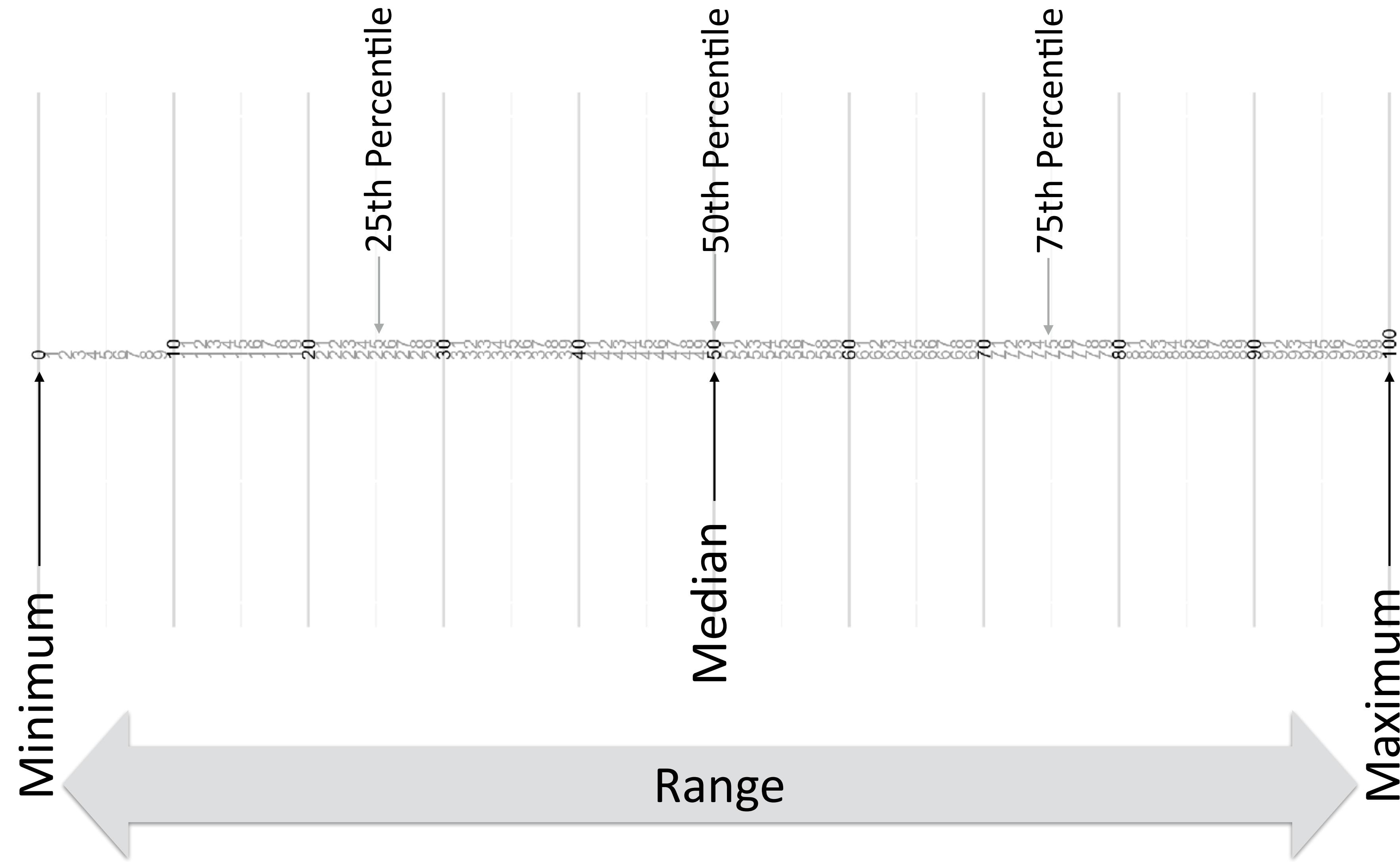
# Single Variable



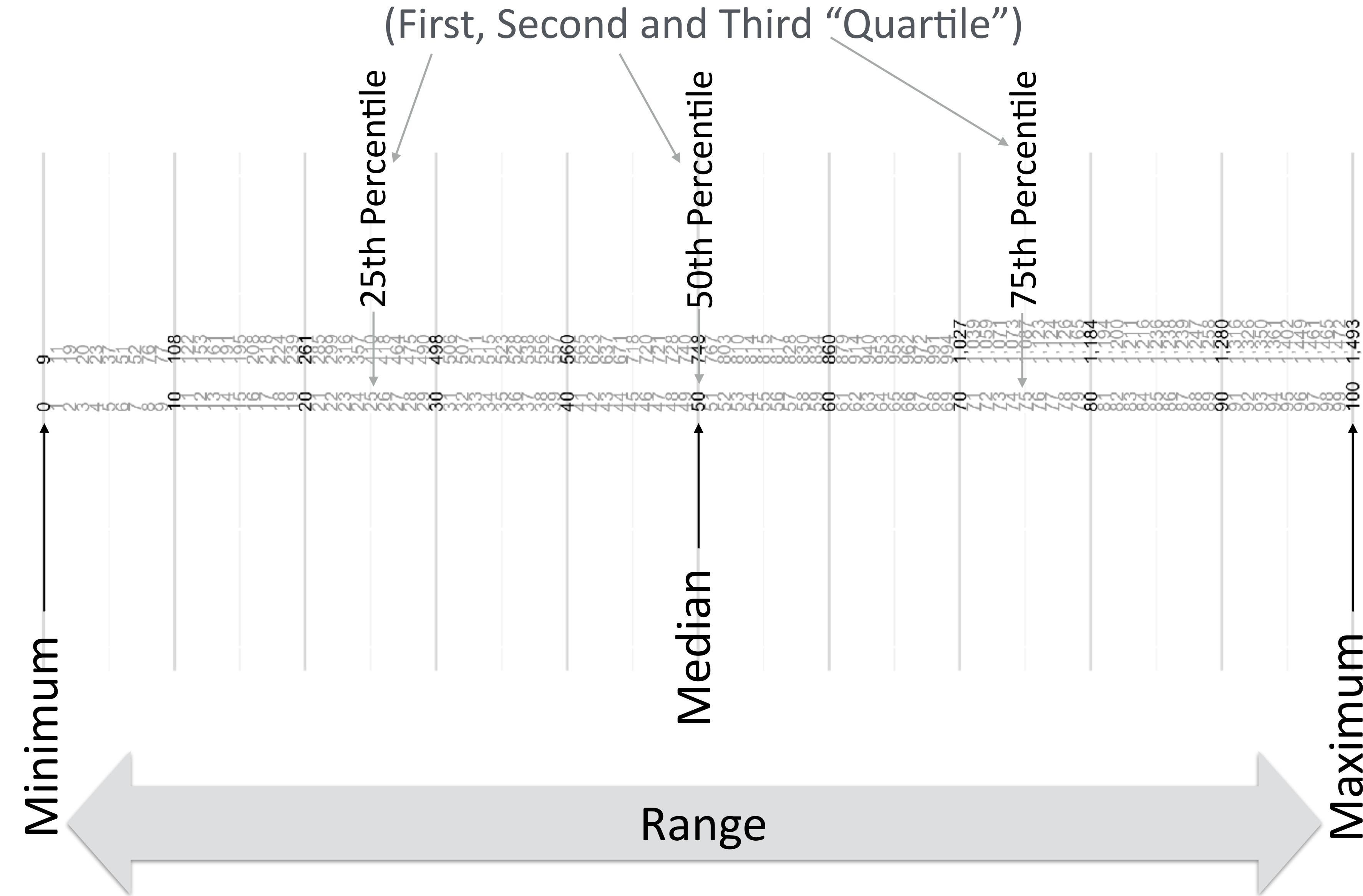
# Single Variable



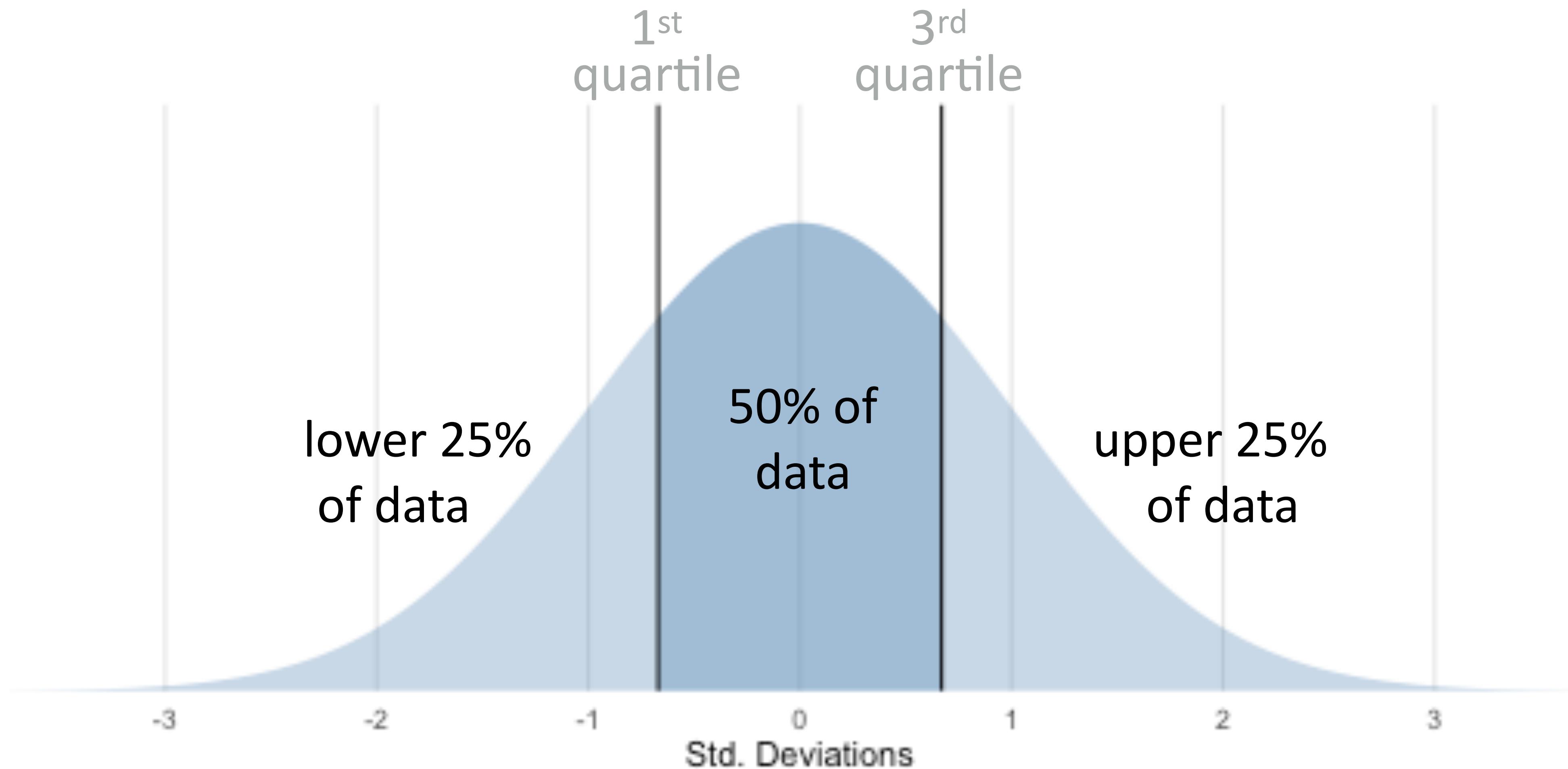
# Single Variable



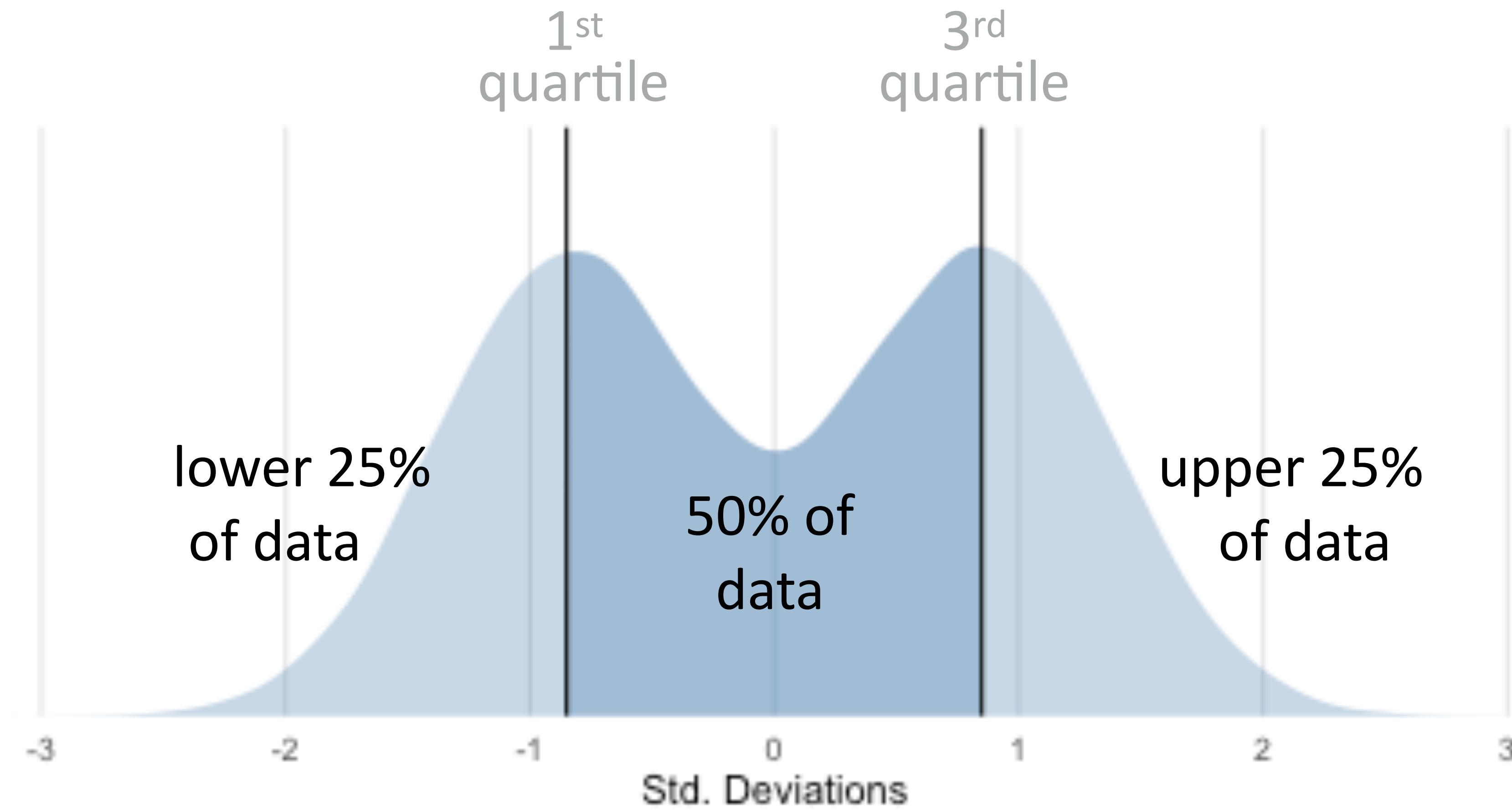
# Five Number Summary



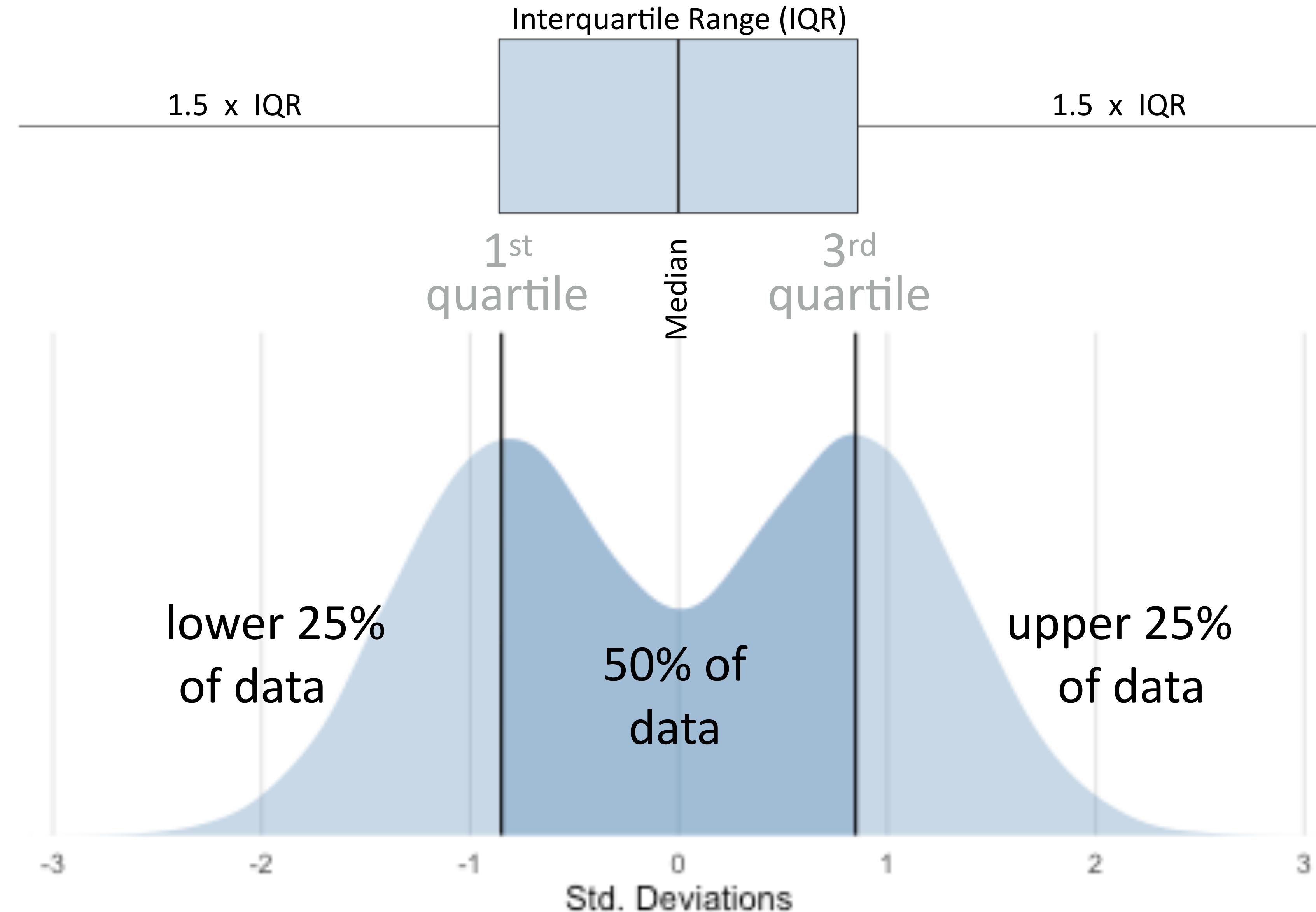
# Visualizing 5-number summaries



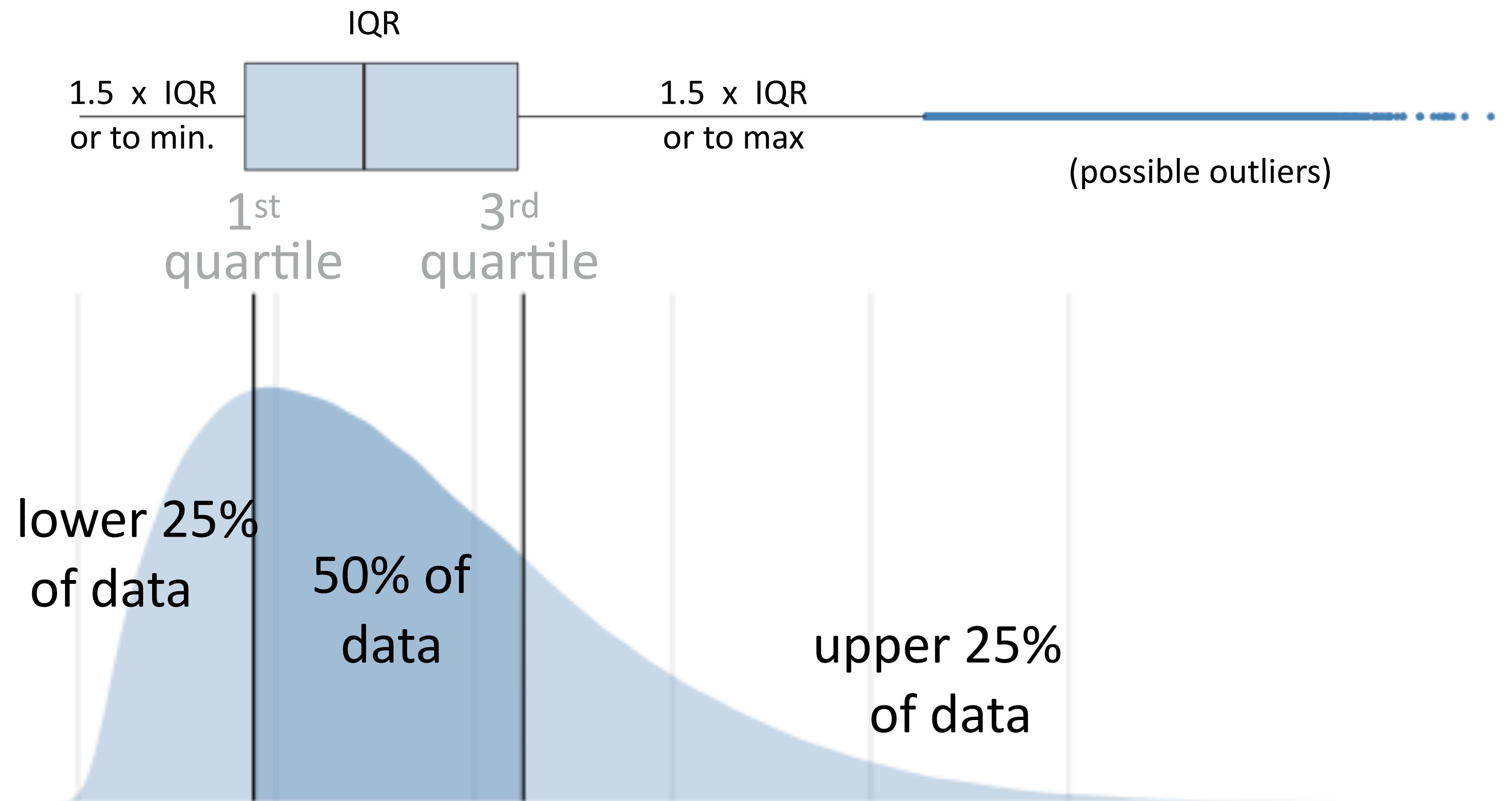
# Bi-Modal (two modes/peaks)



# Bi-Modal (two modes/peaks)

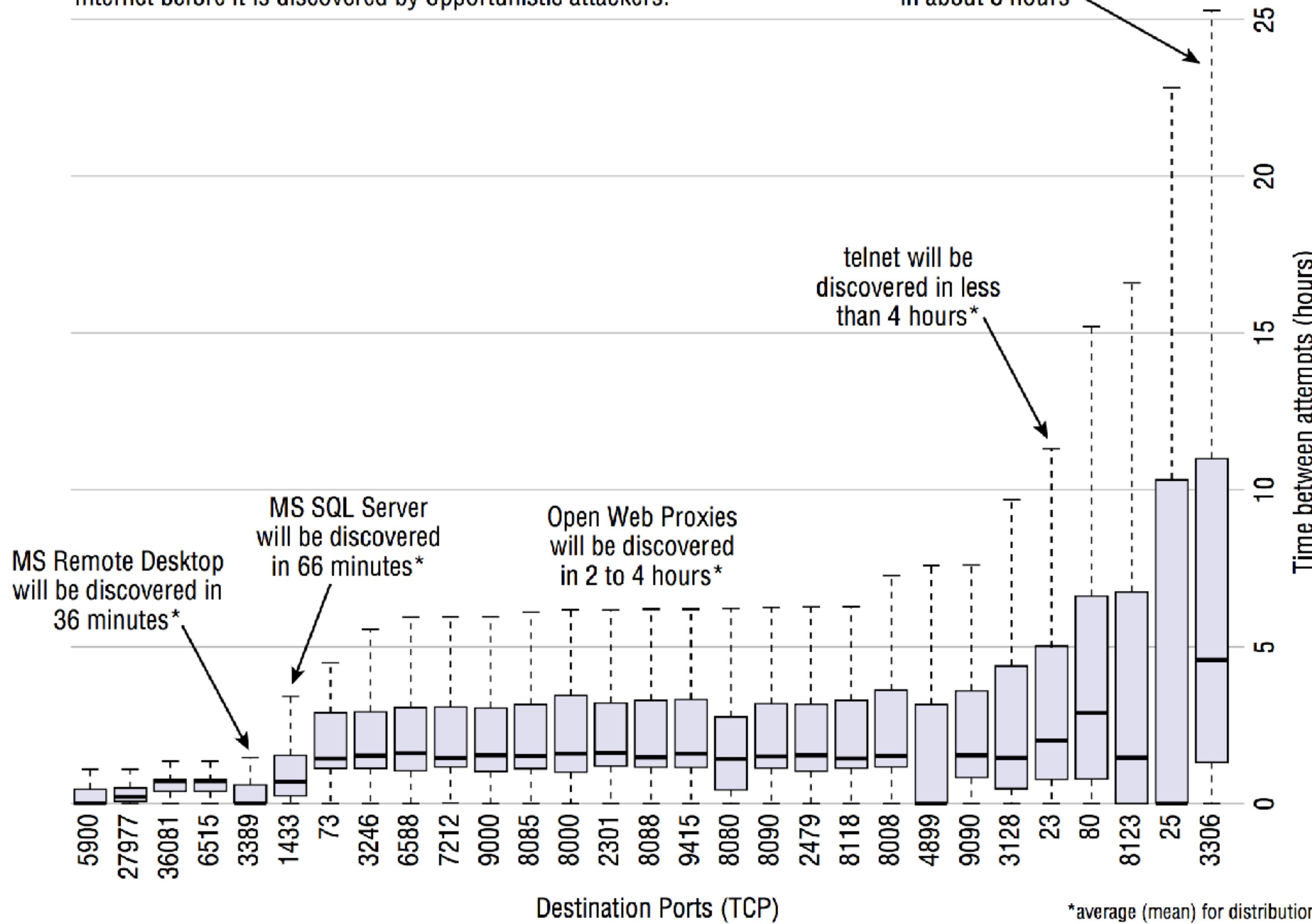


# 5-Number Summaries with Skew

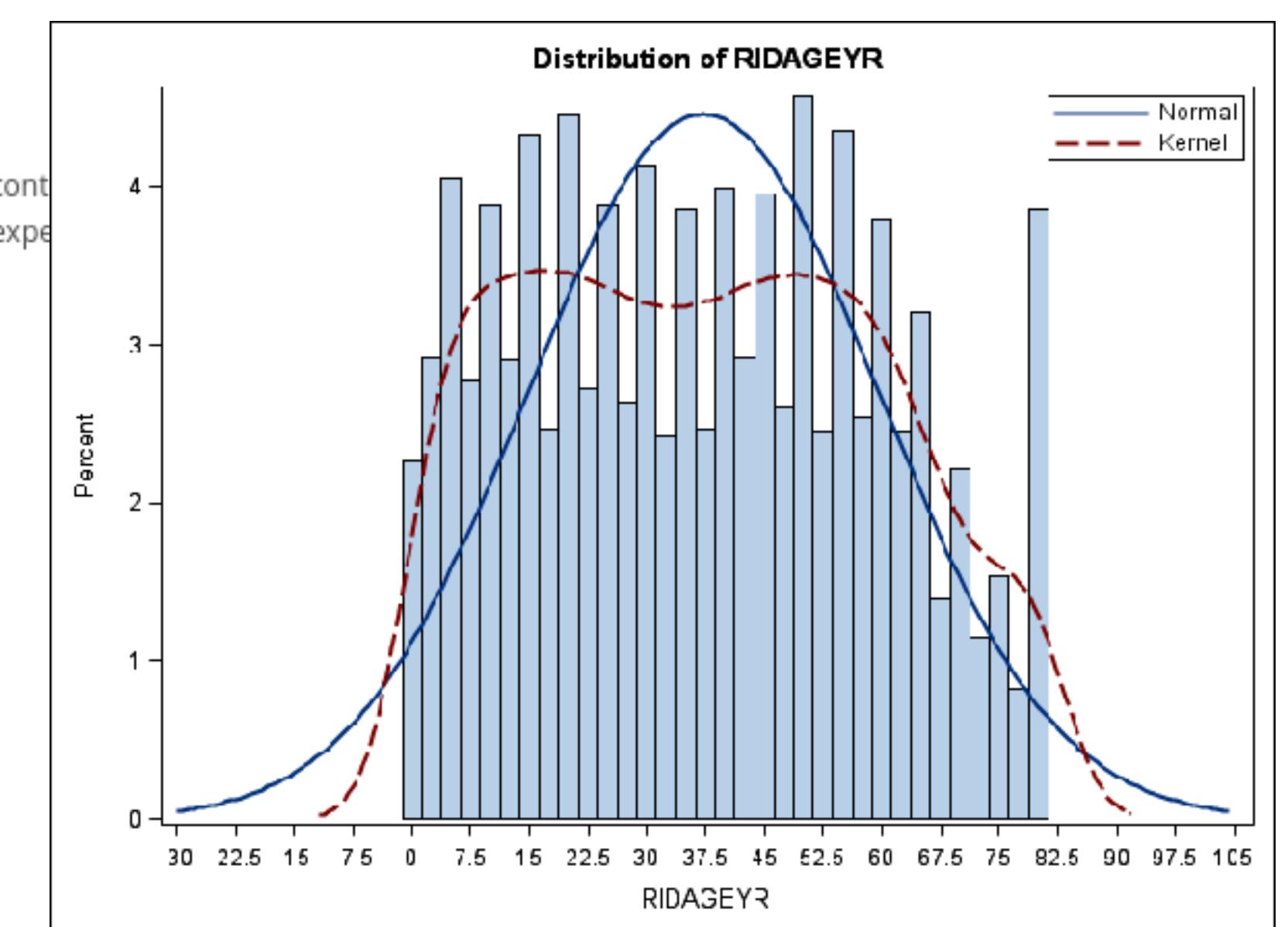
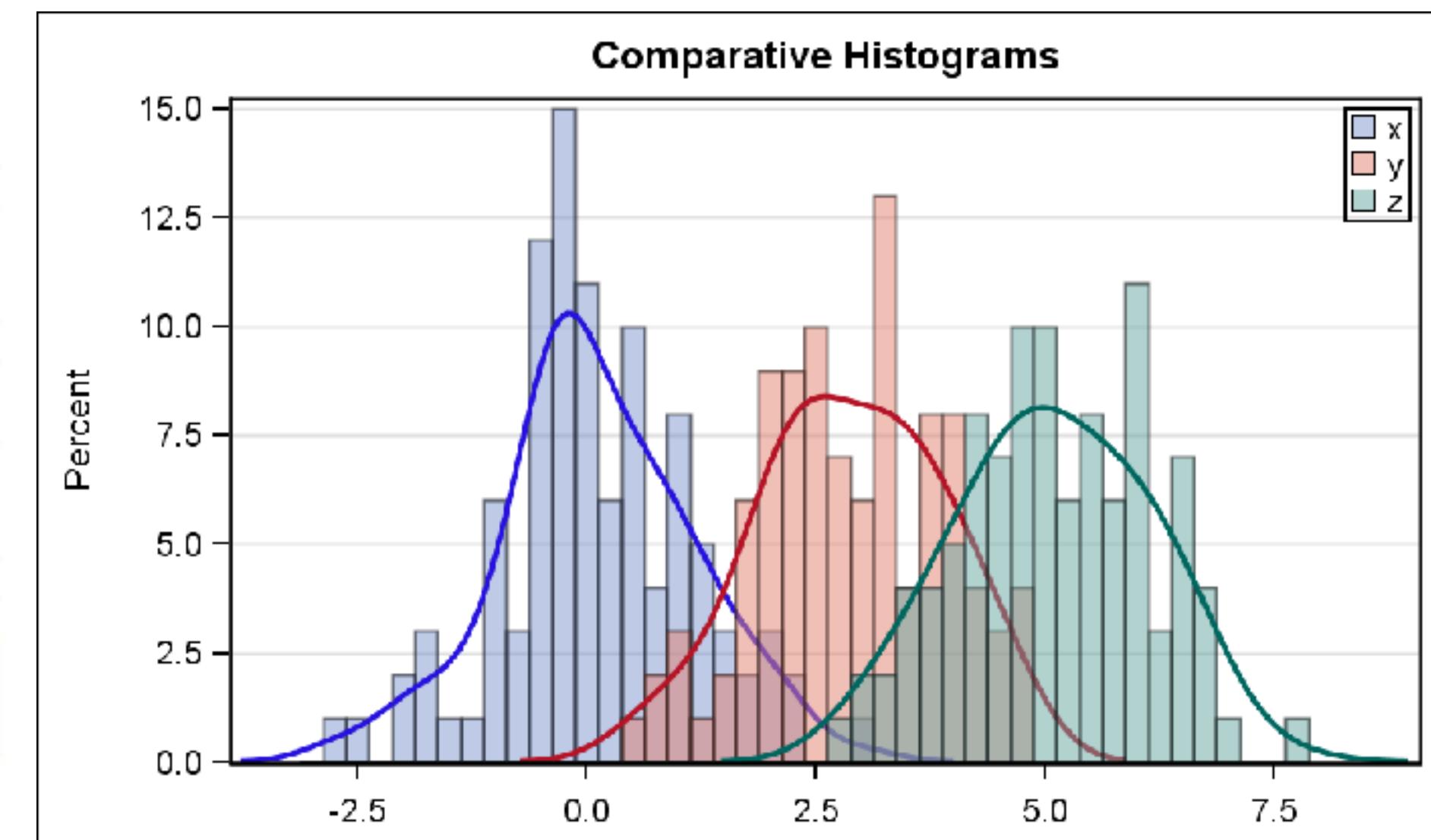
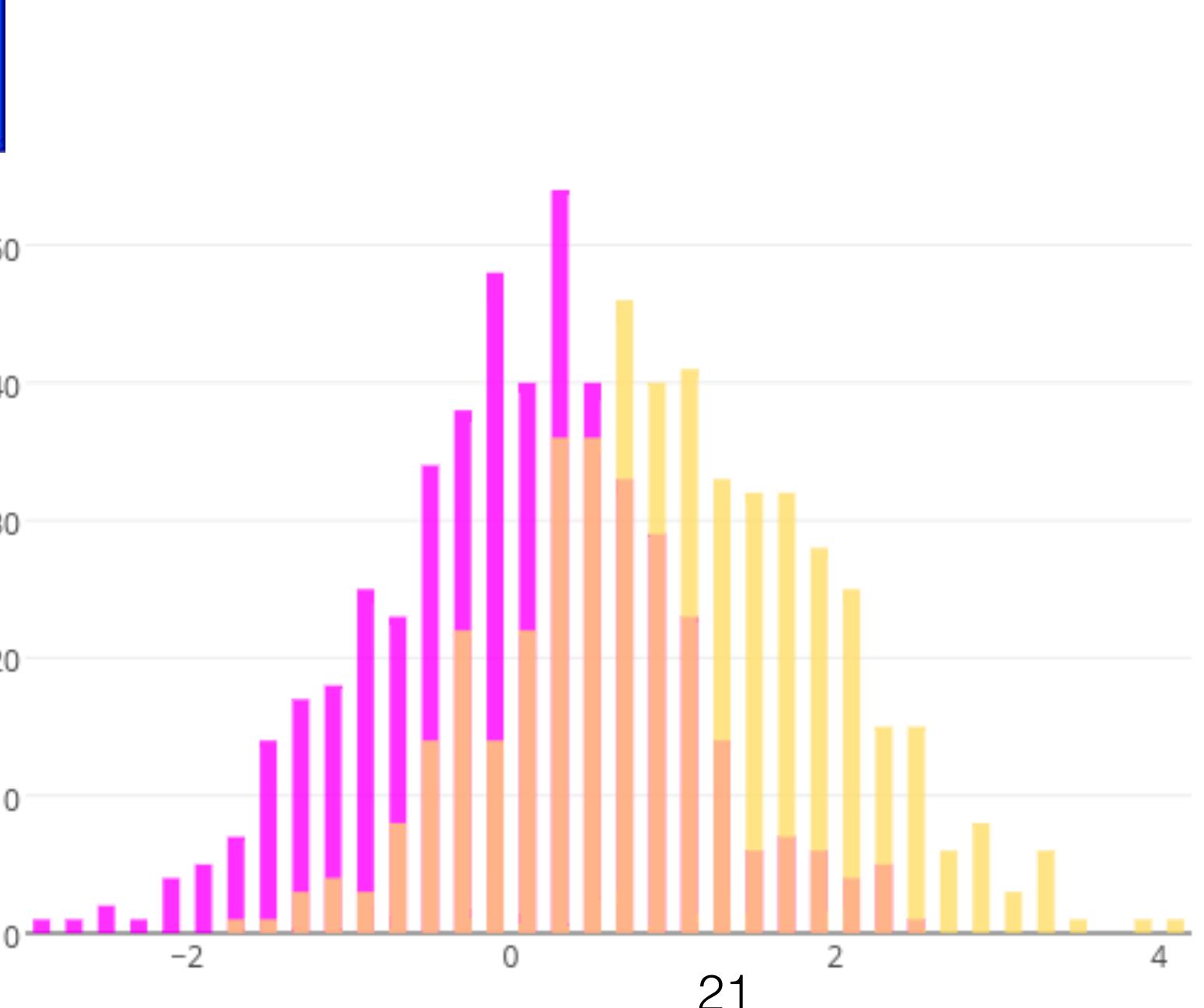
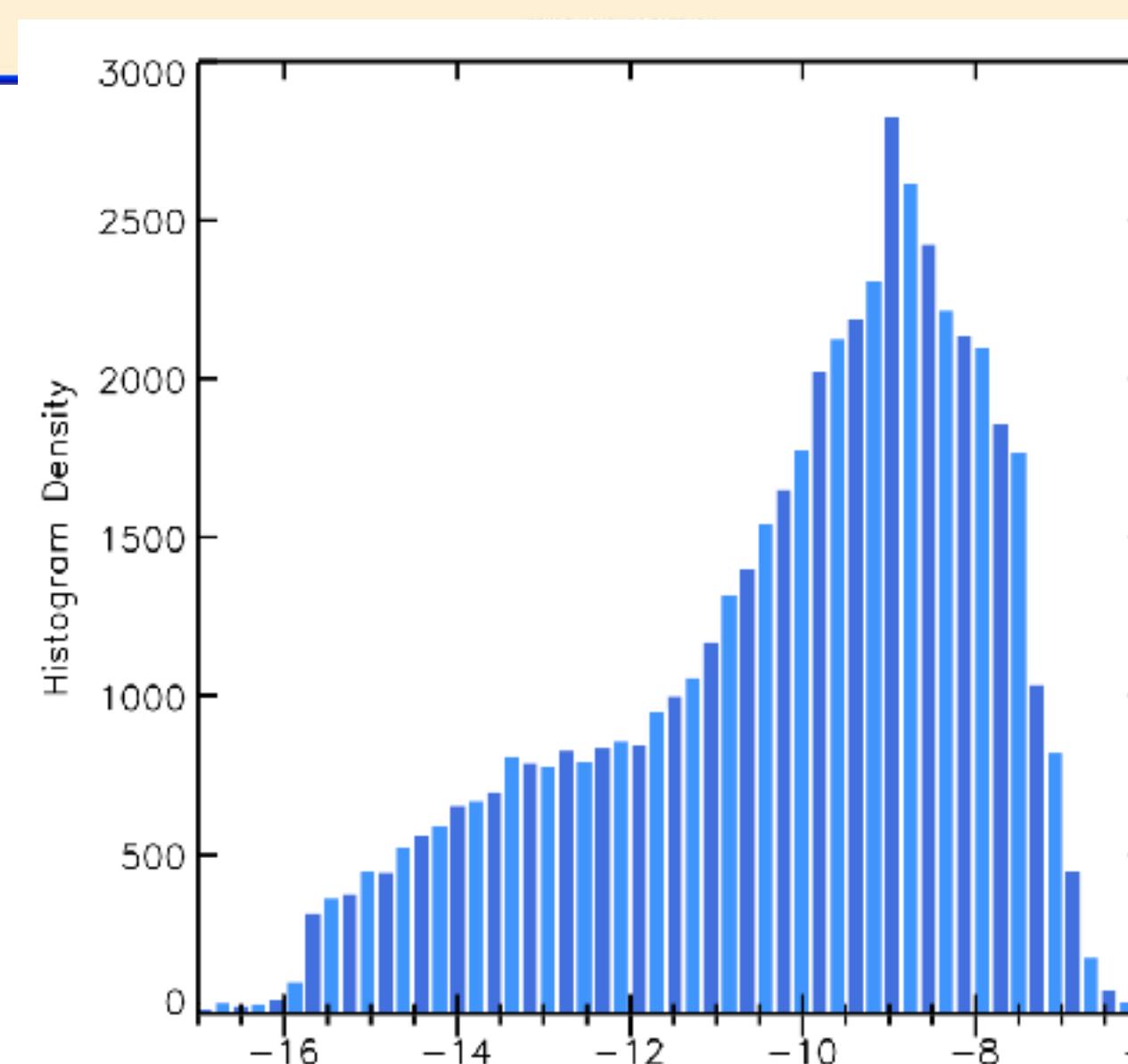
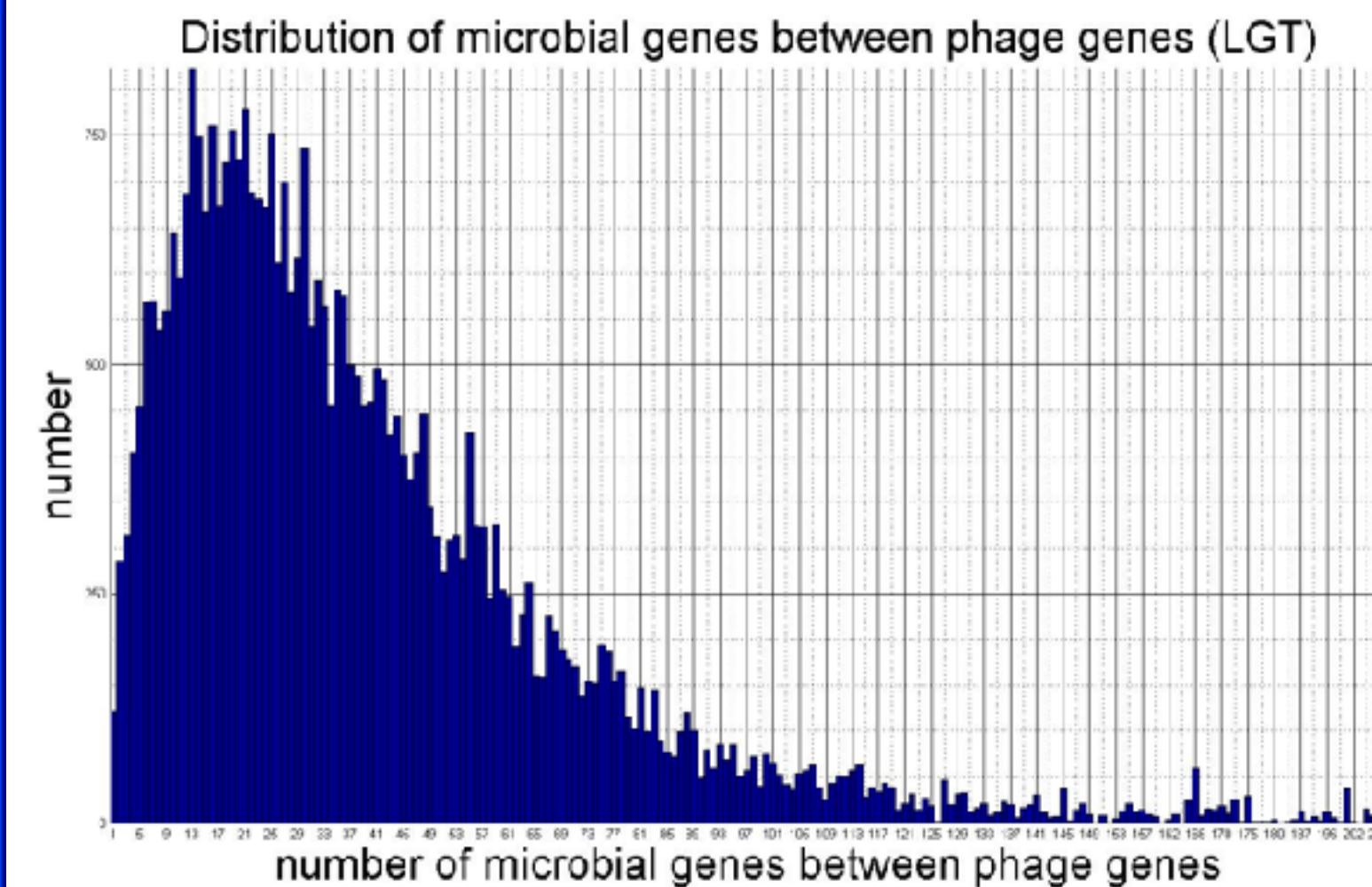
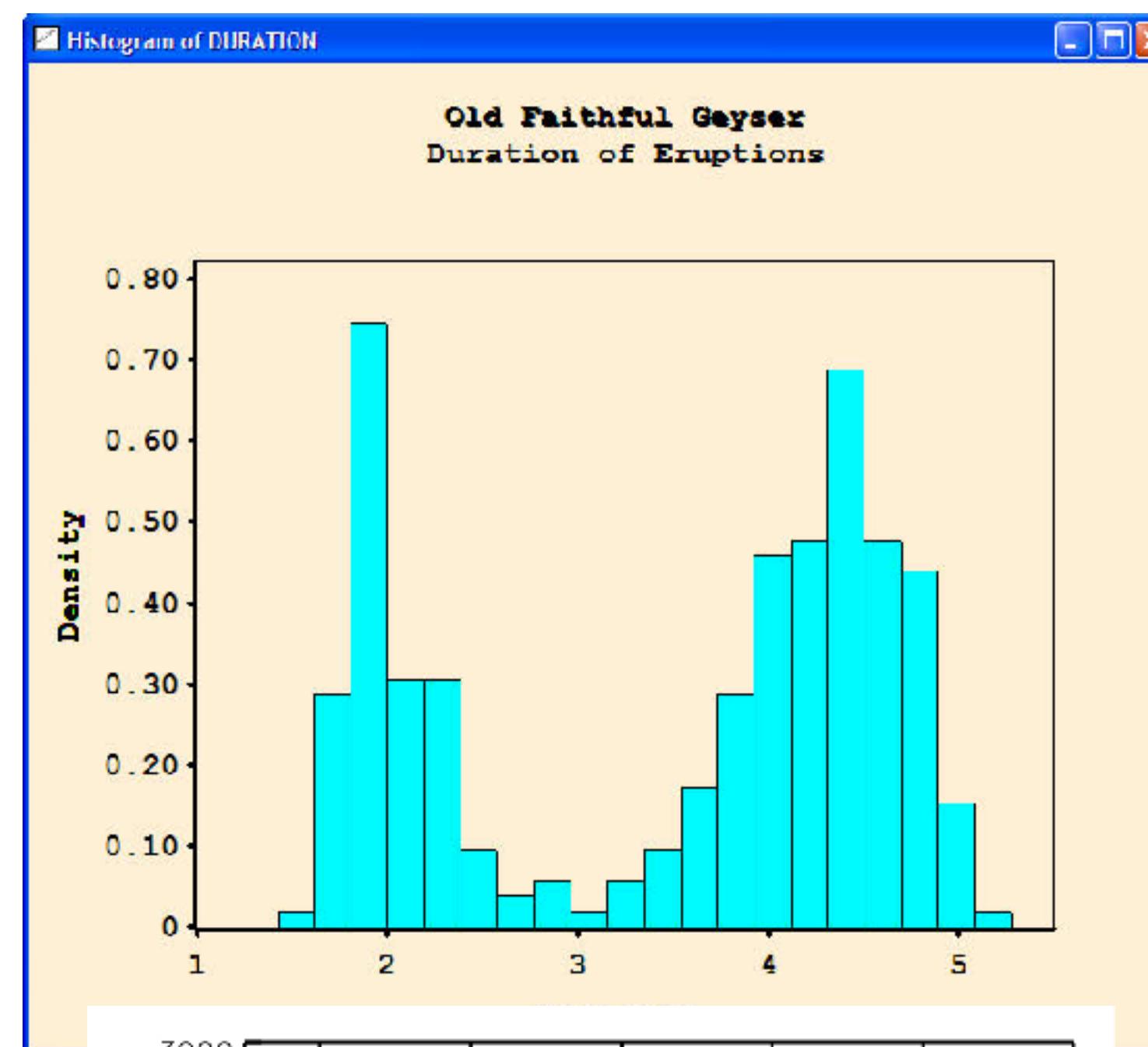


## How long will a service go undiscovered by opportunistic attackers?

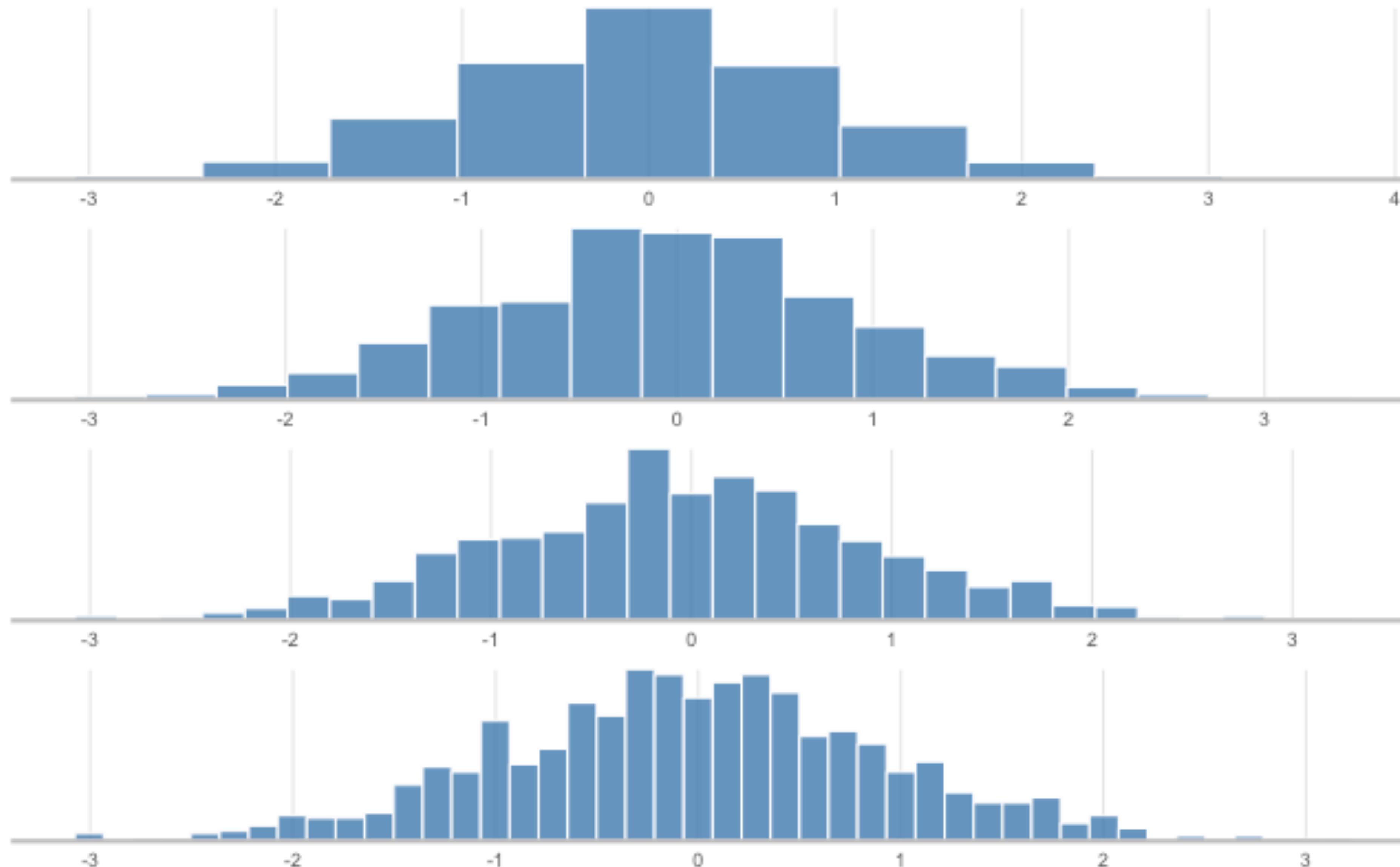
By measuring the time between packets received, it is possible to estimate how long a service may be on the Internet before it is discovered by opportunistic attackers.



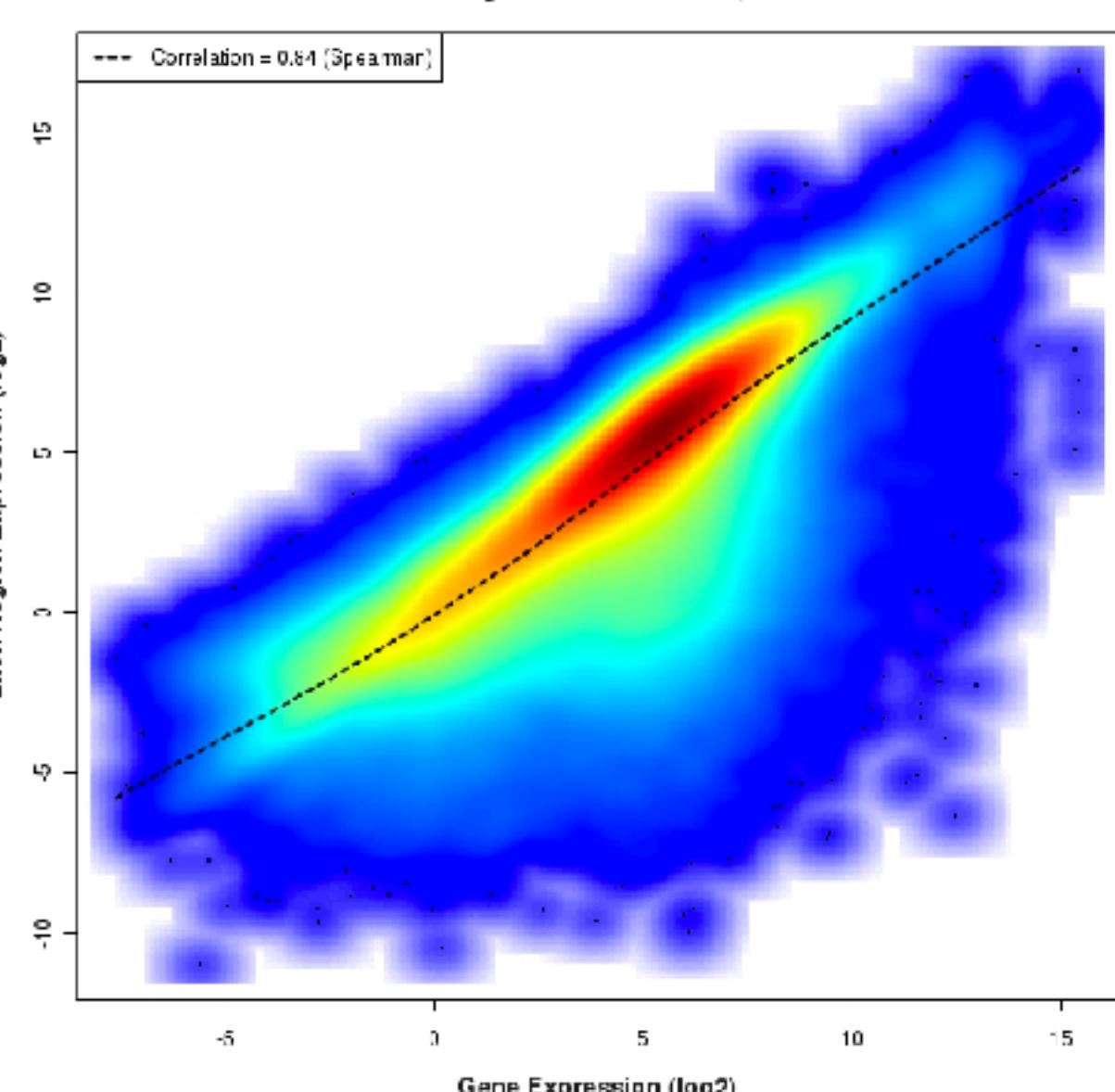
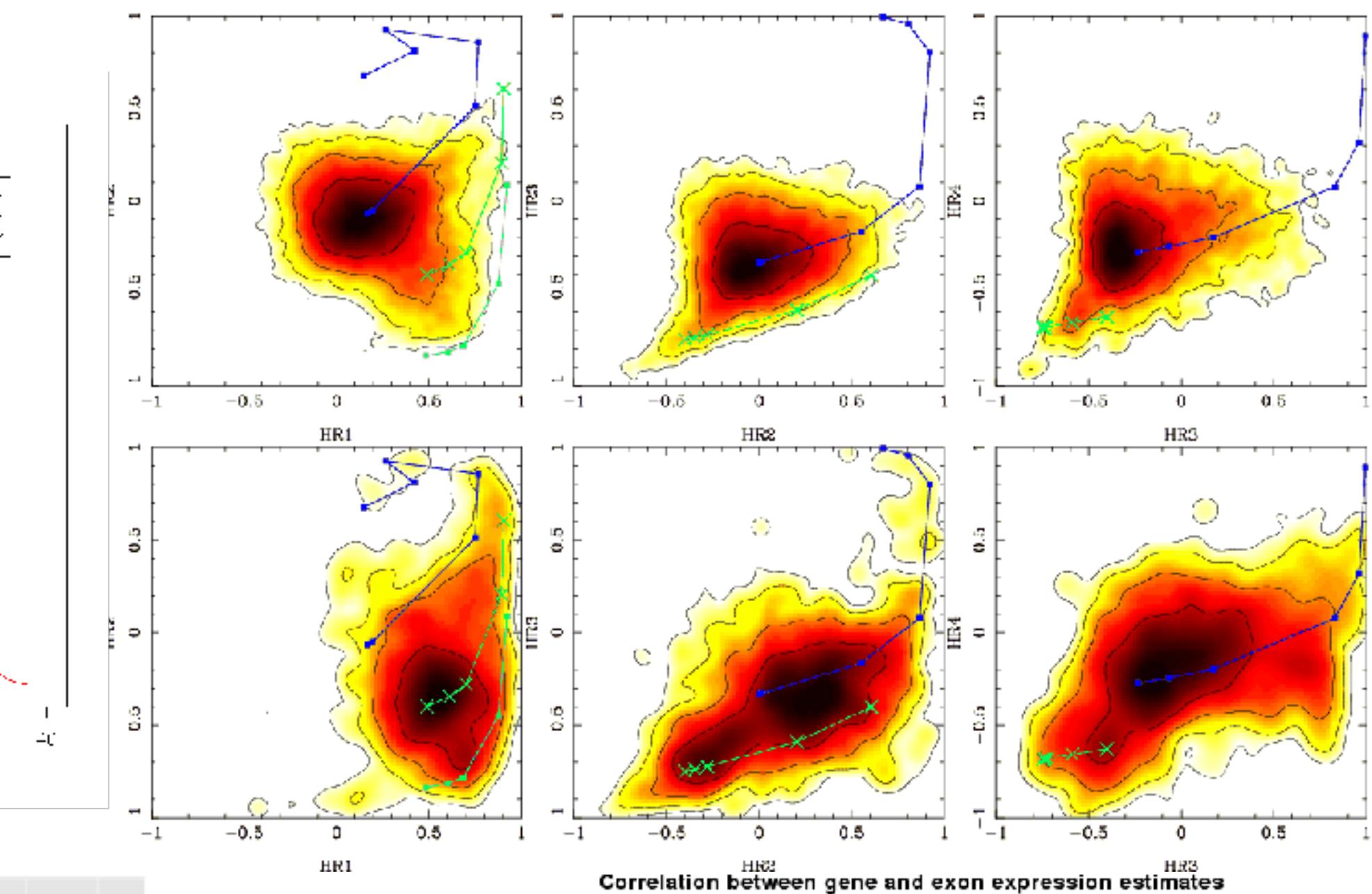
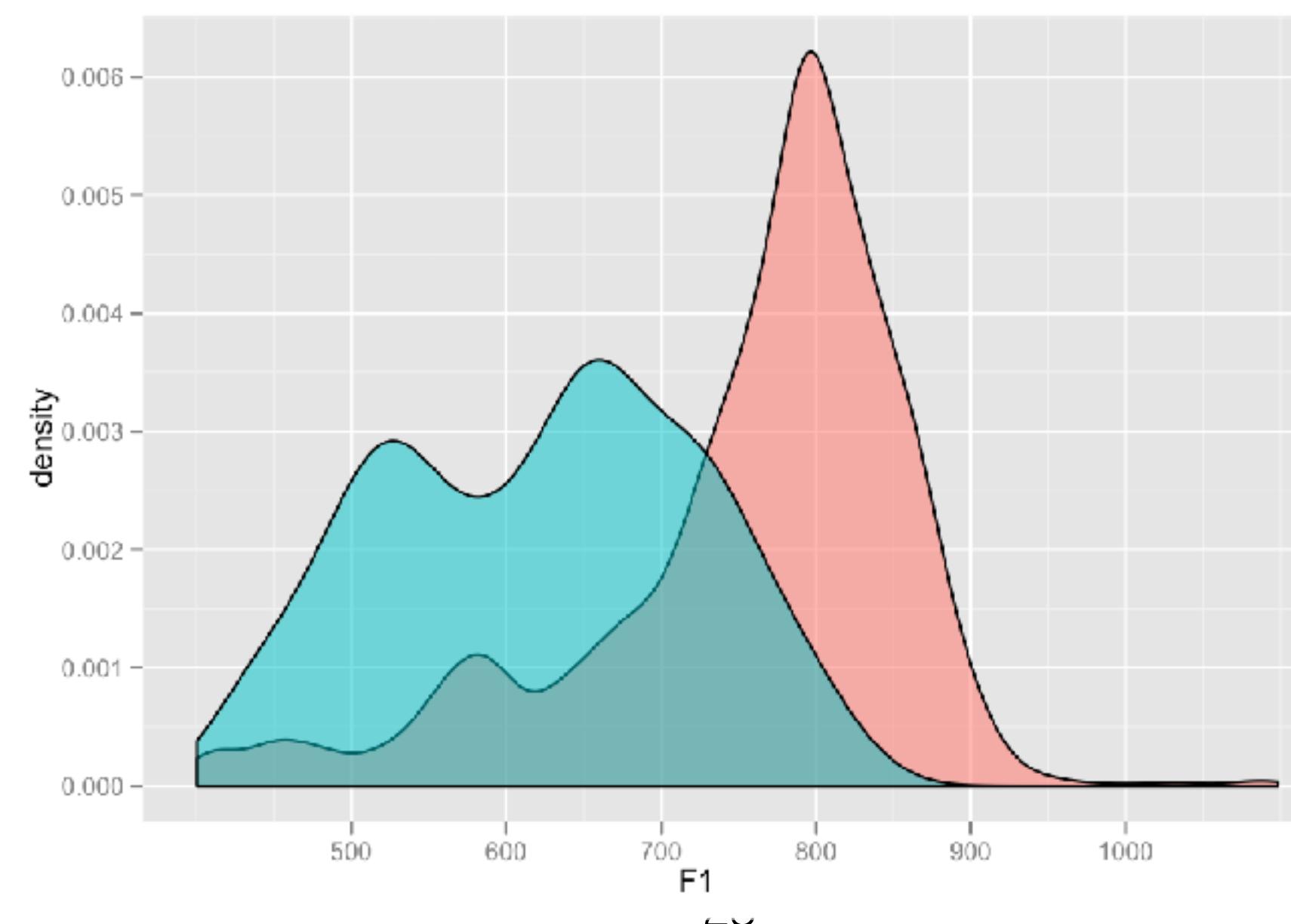
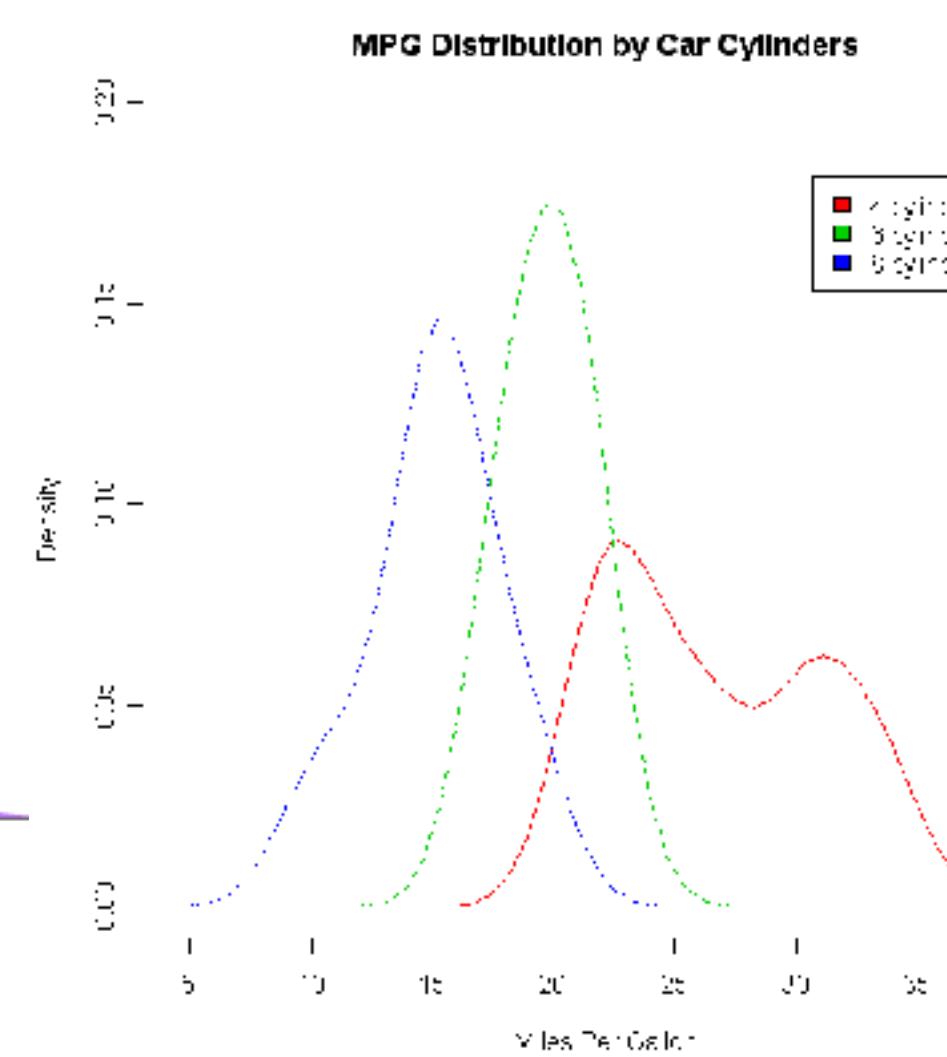
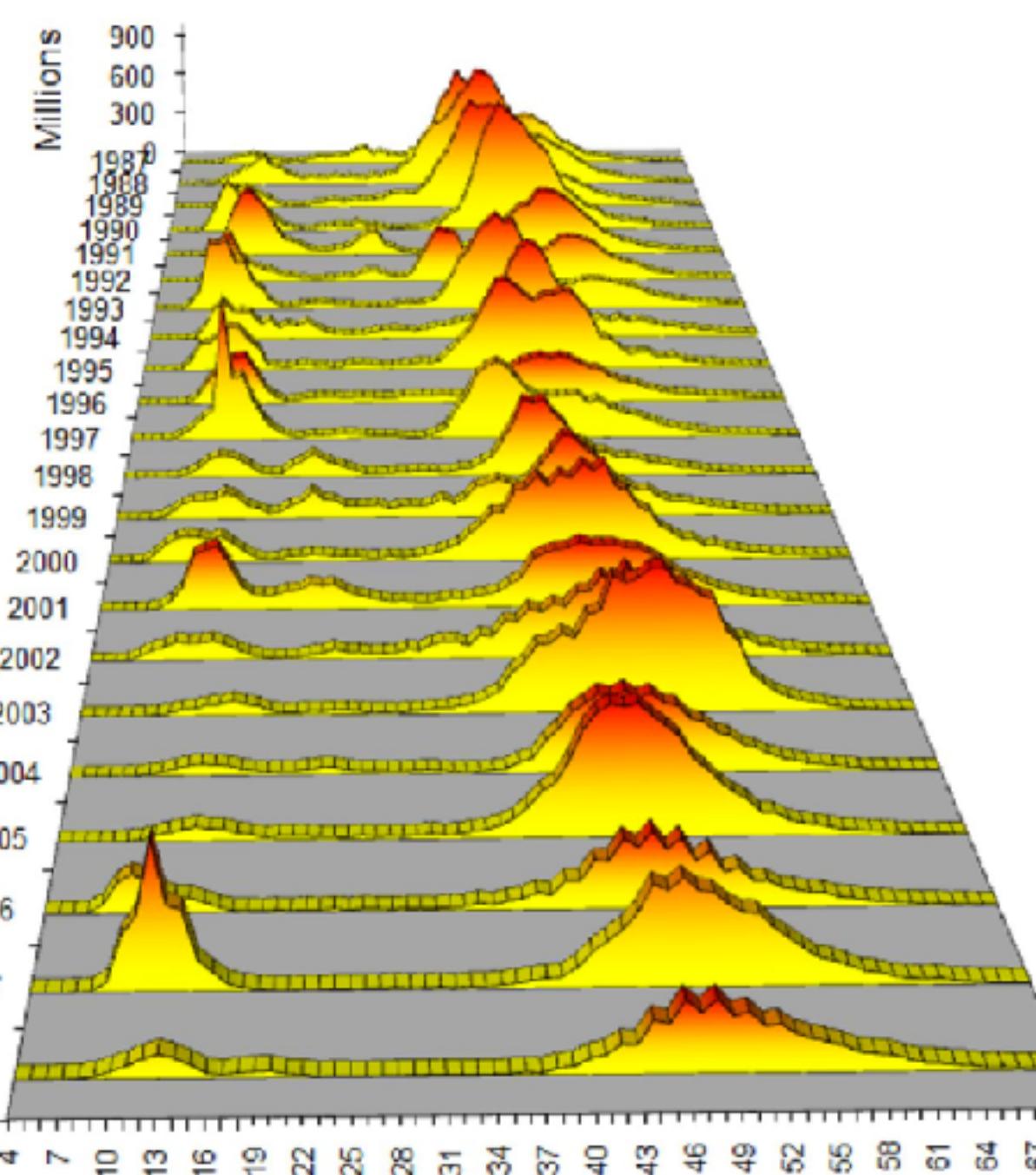
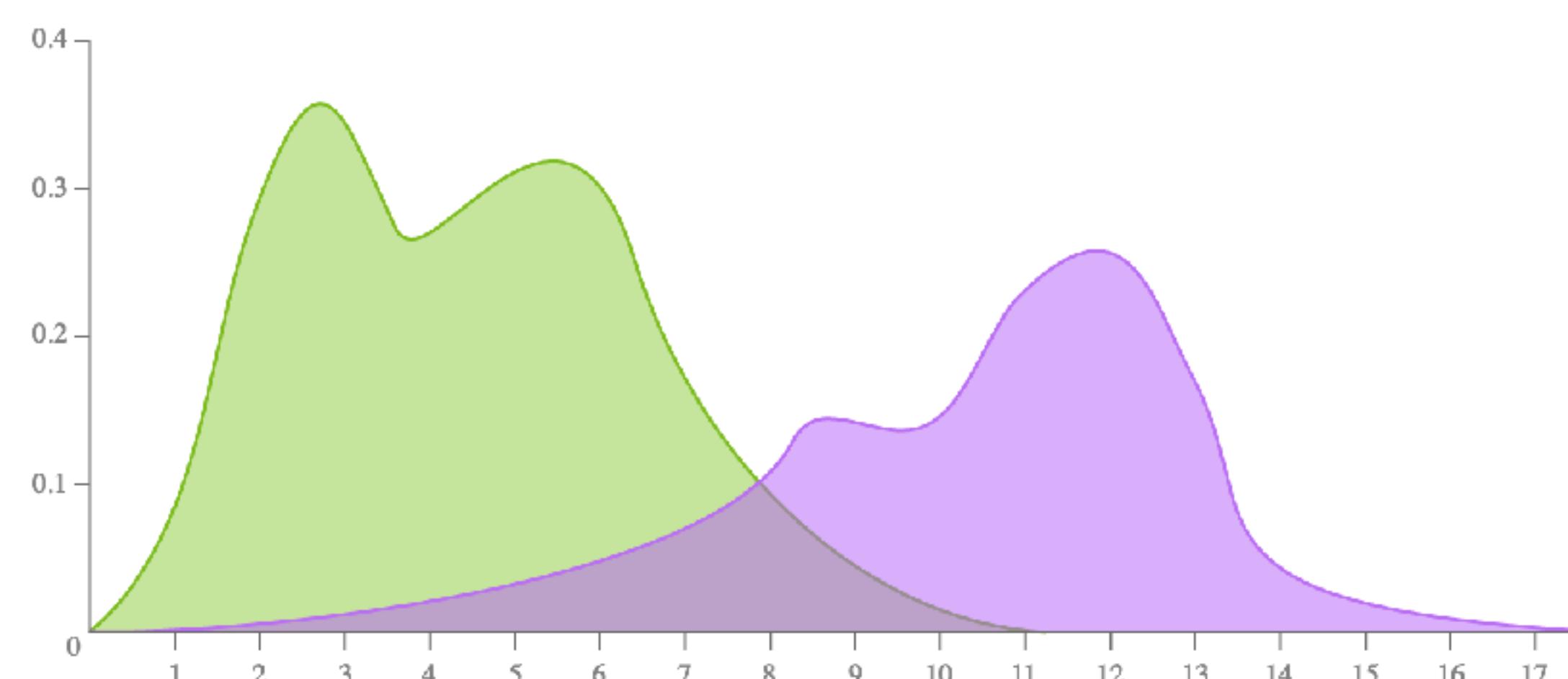
# Visualizing Distributions: Histogram



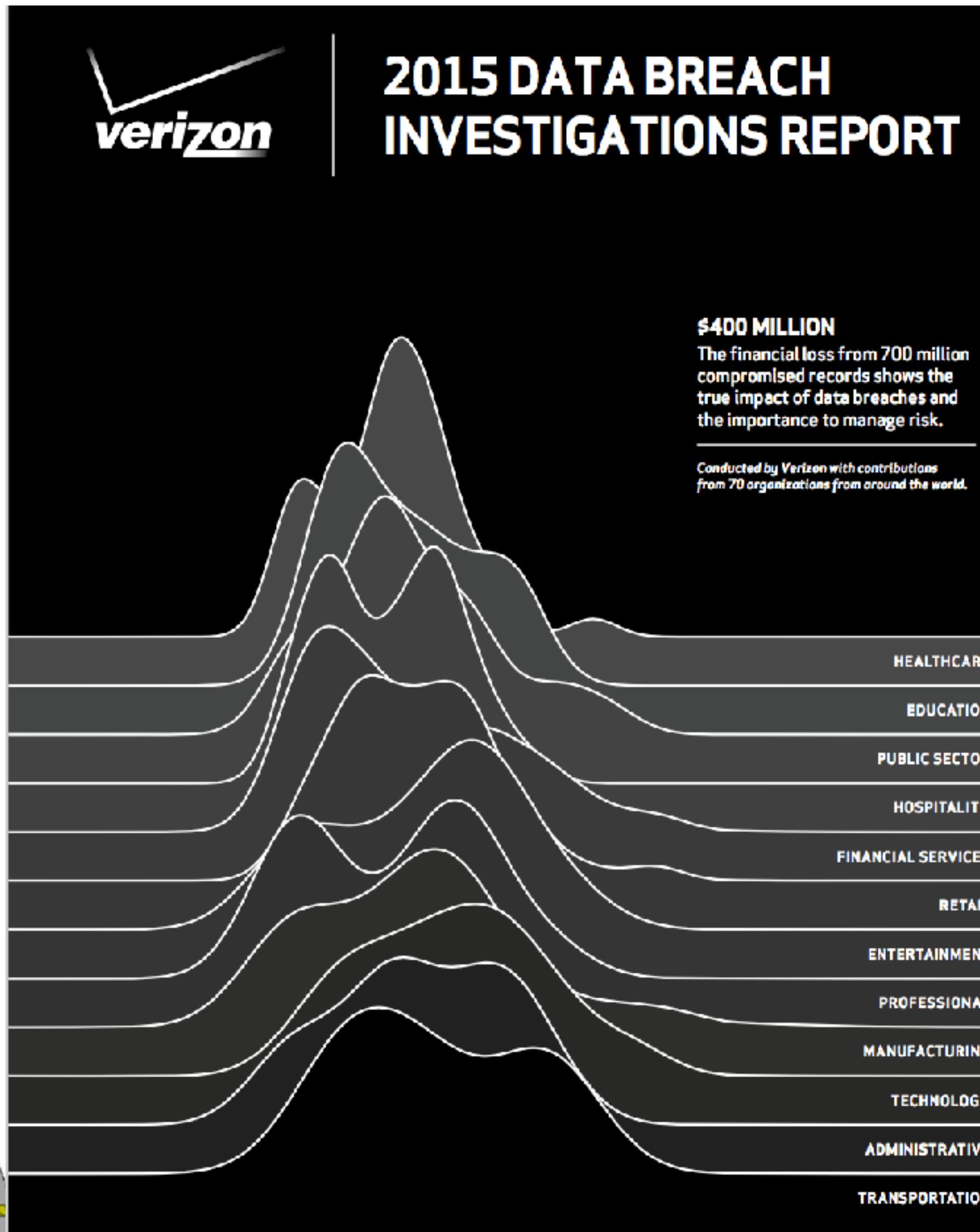
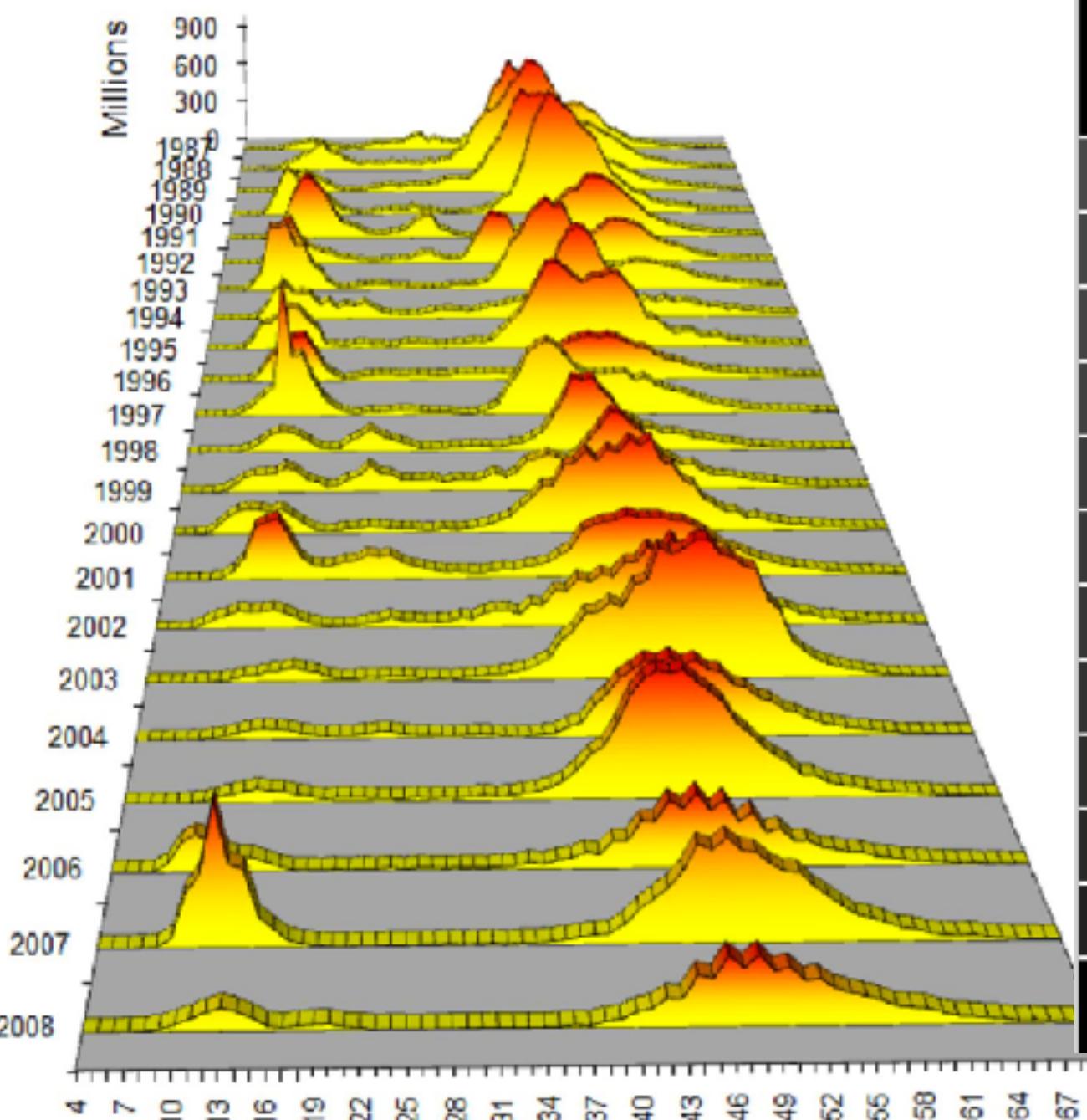
# Visualizing Distributions: Histogram



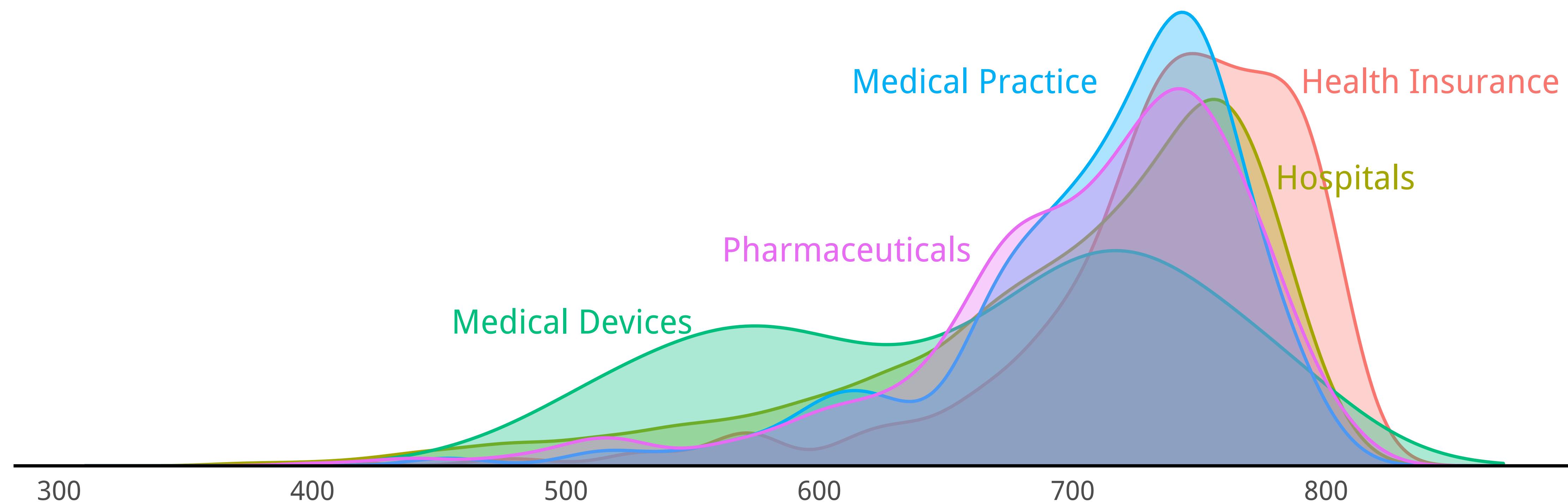
# Visualizing Distributions: Density plots



# “Joy plots”

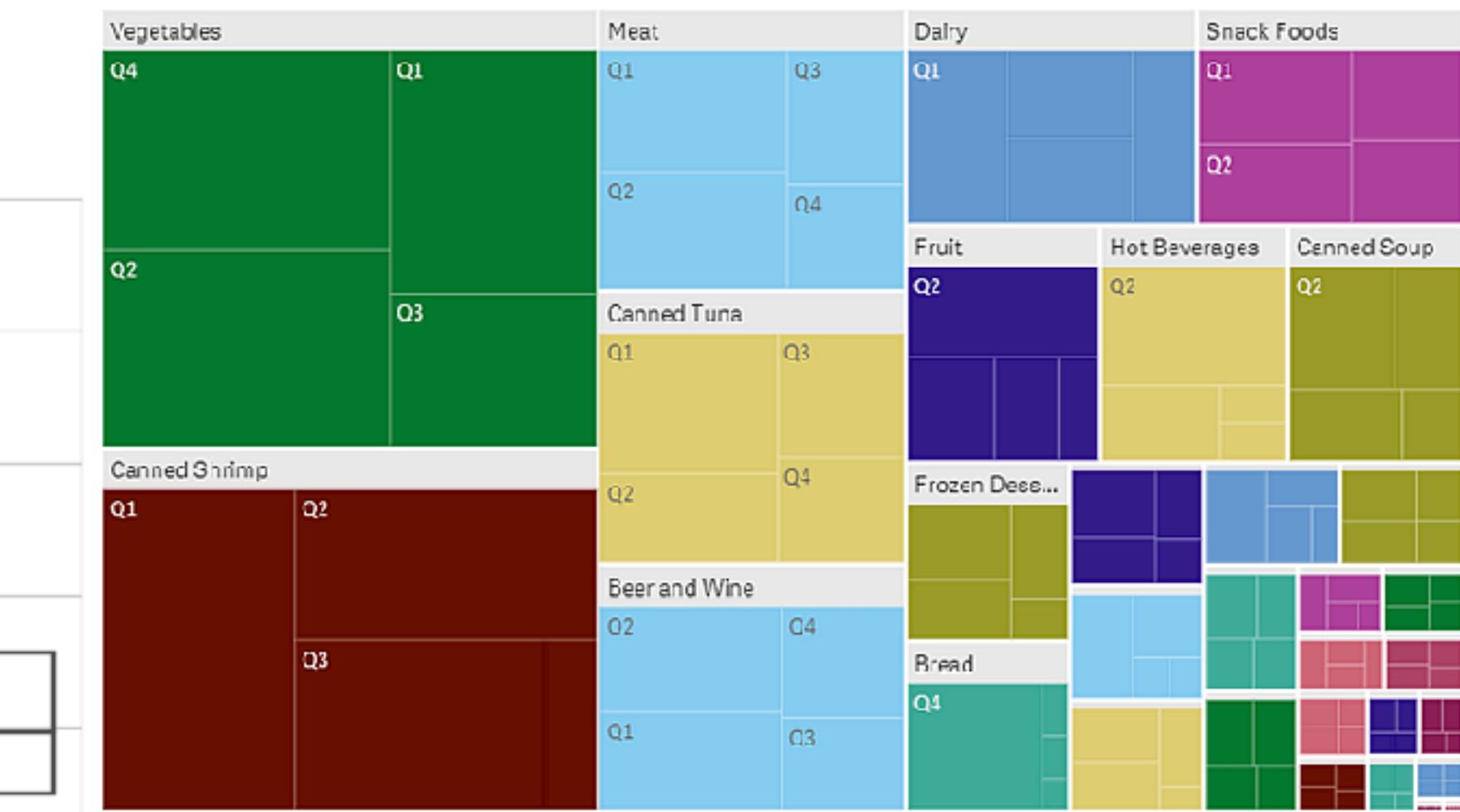
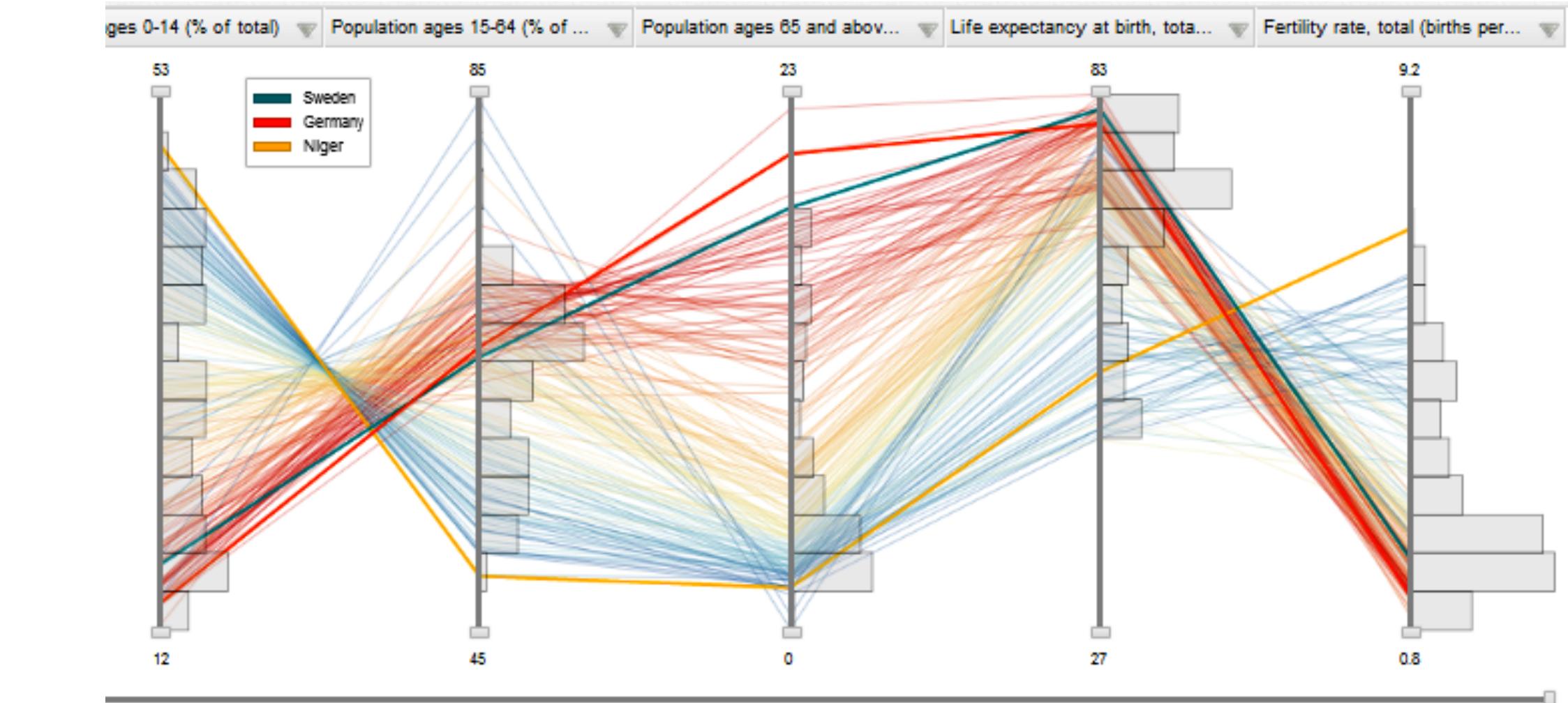
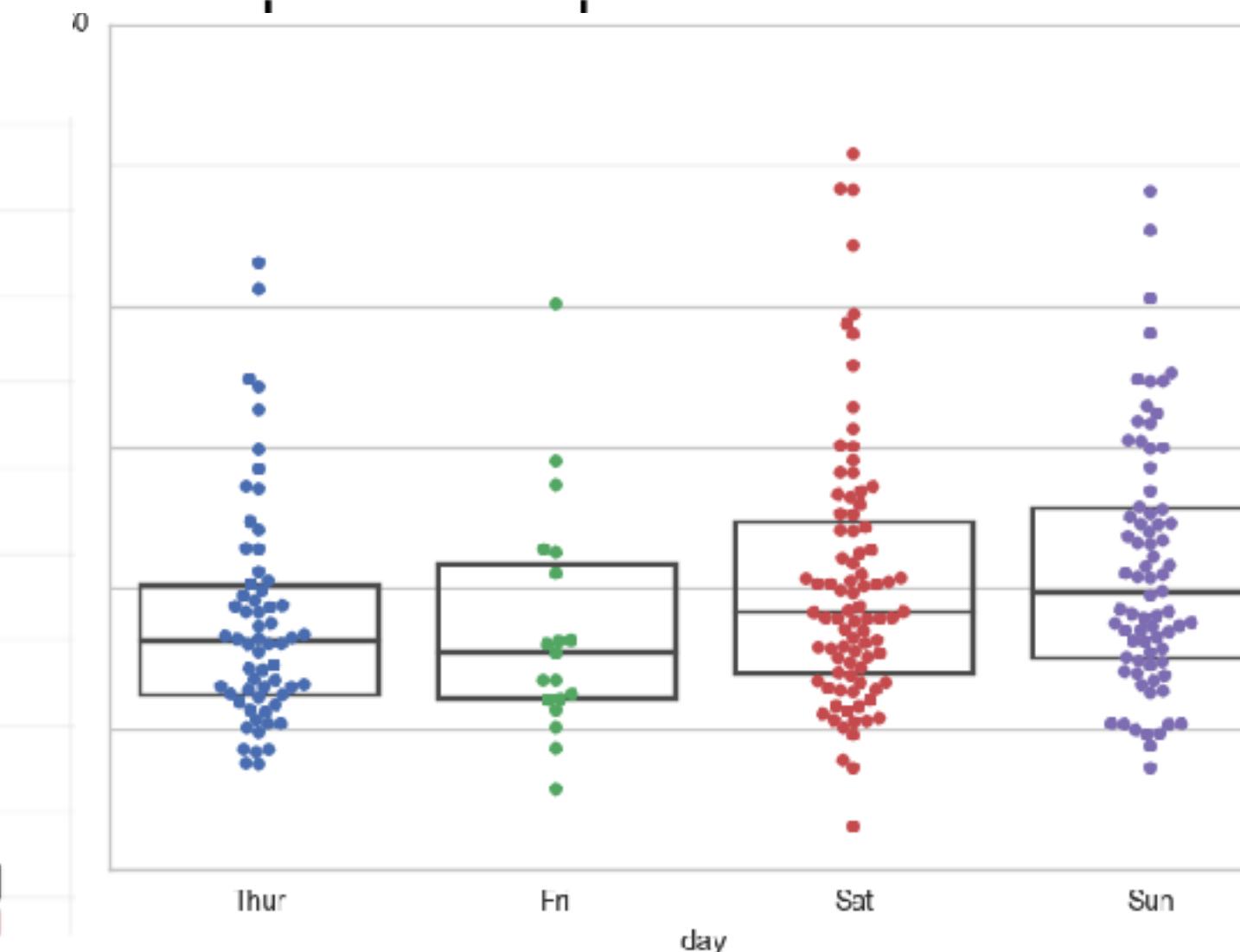
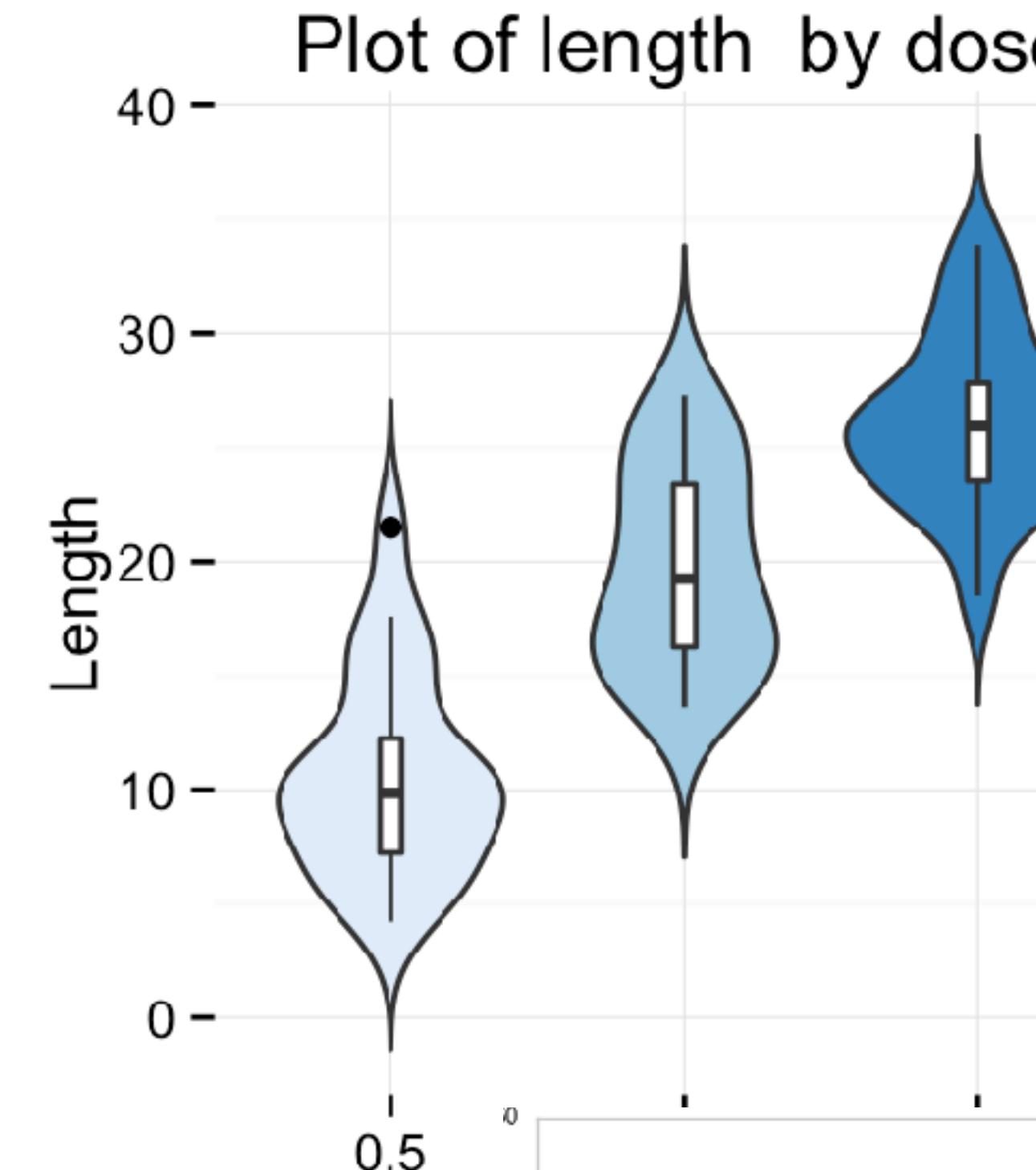
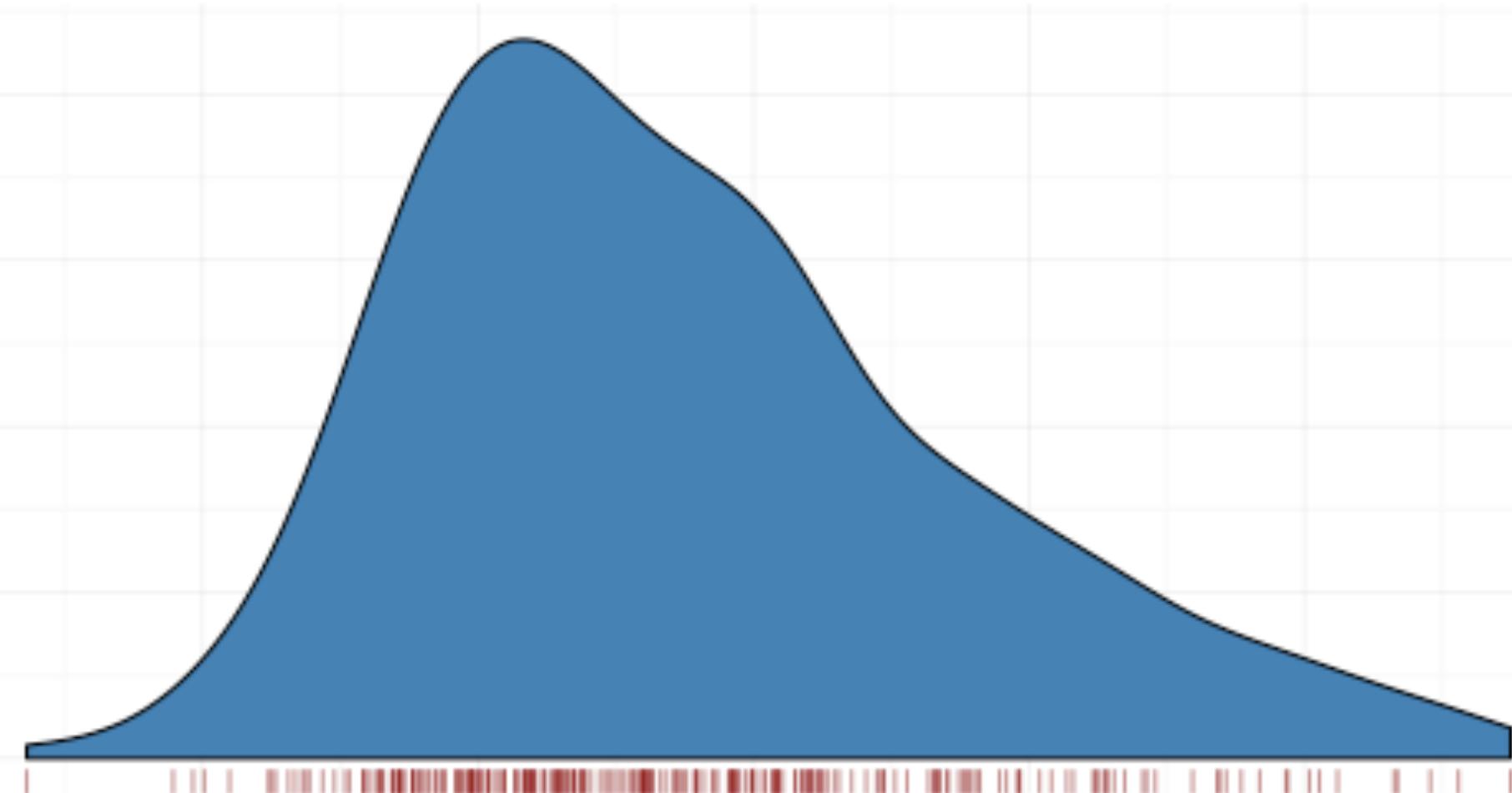
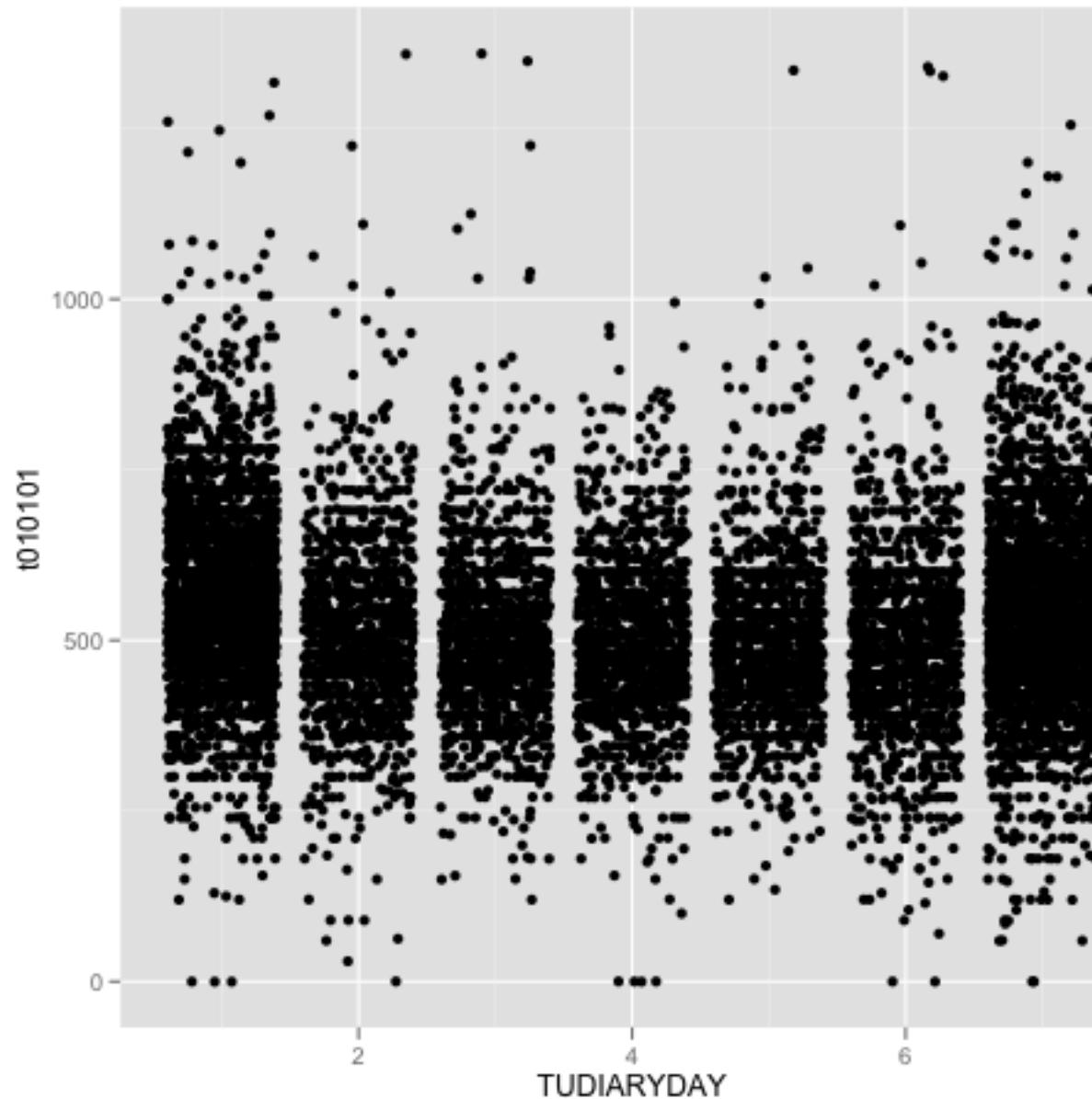


# Visualizing Distributions: Density plots



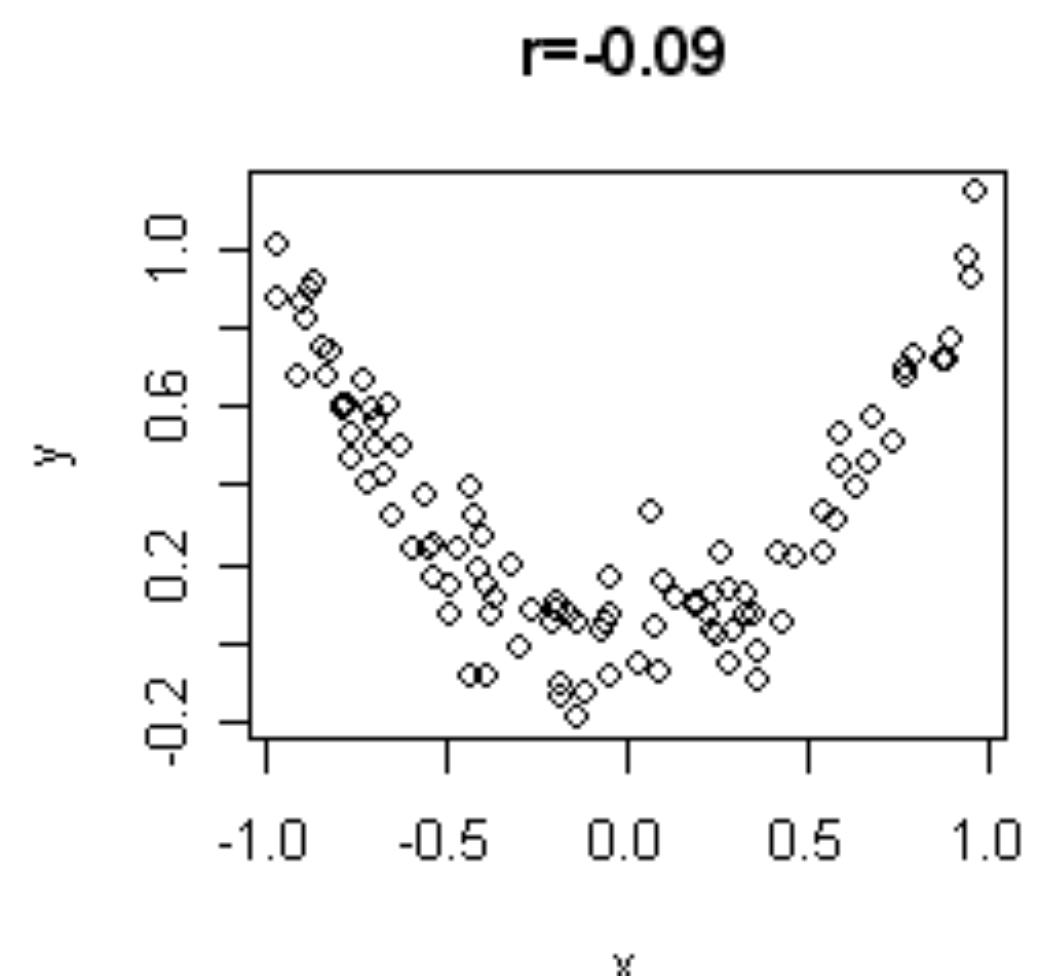
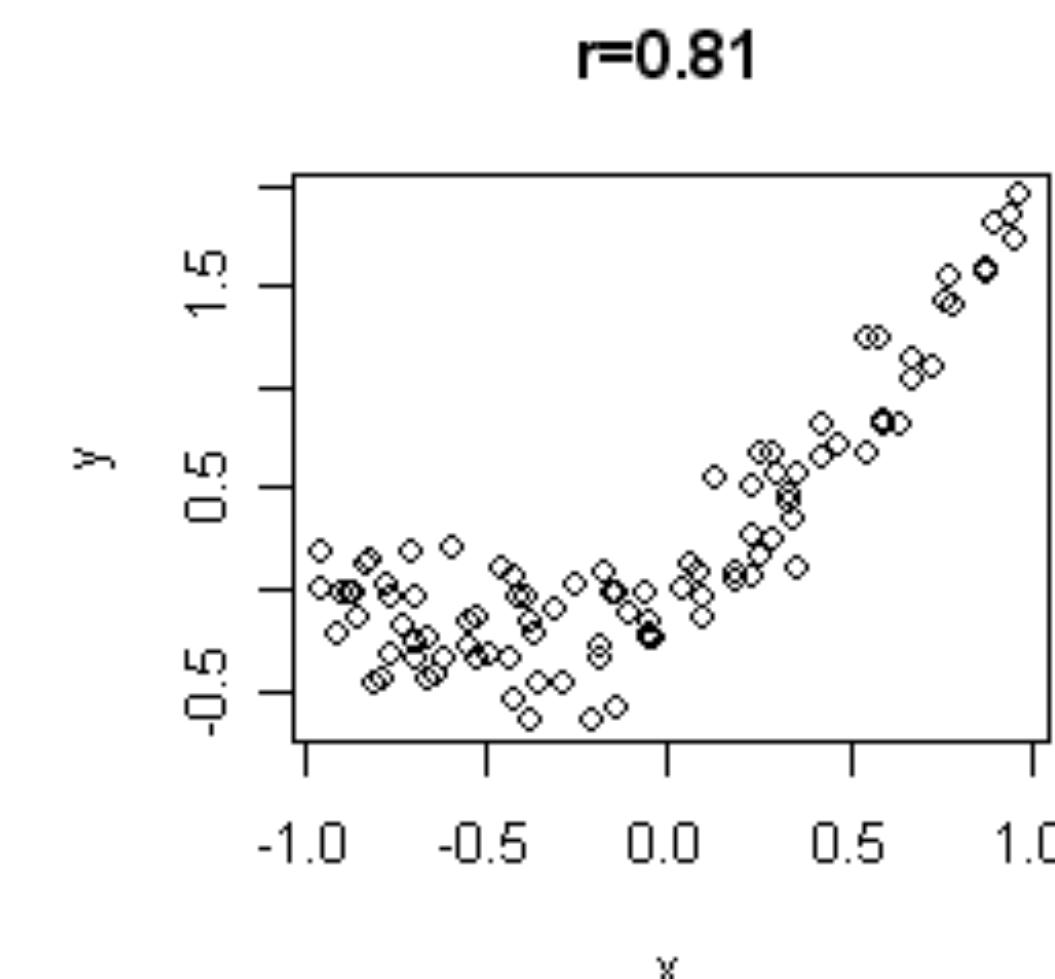
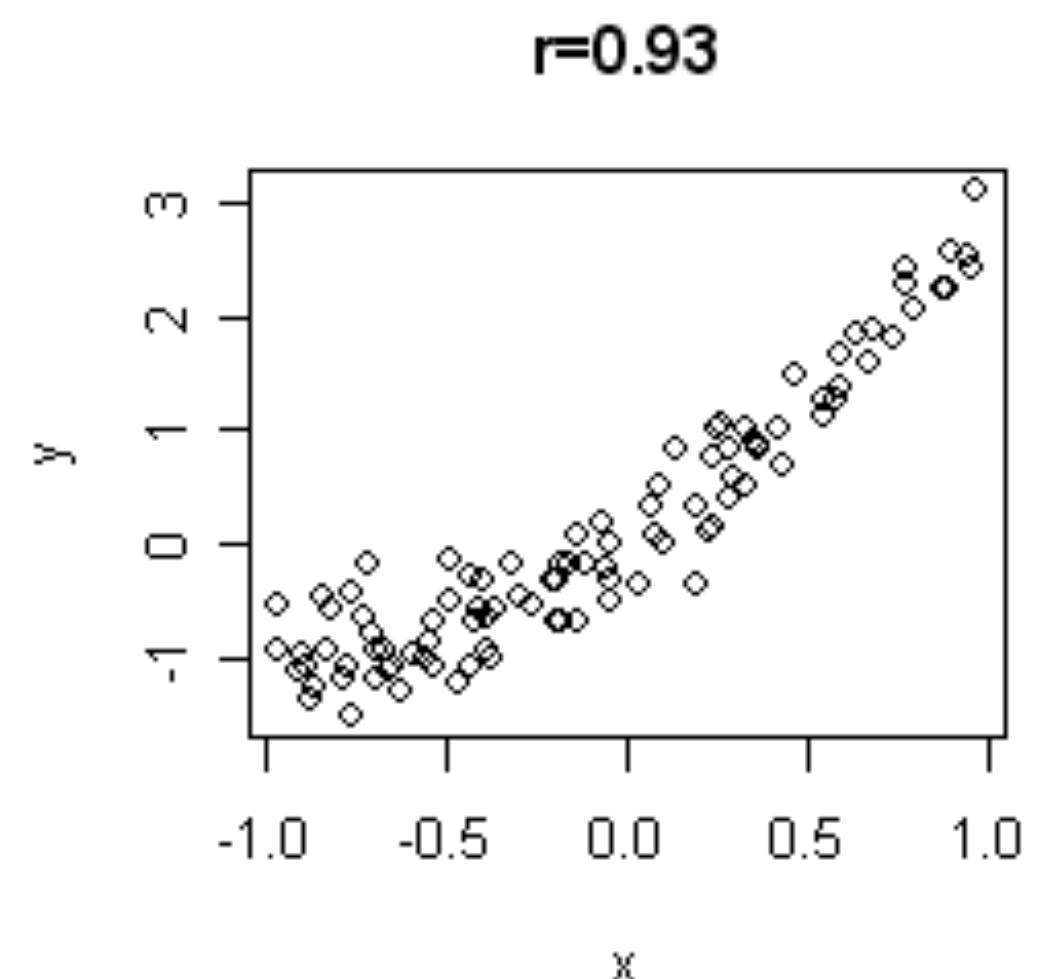
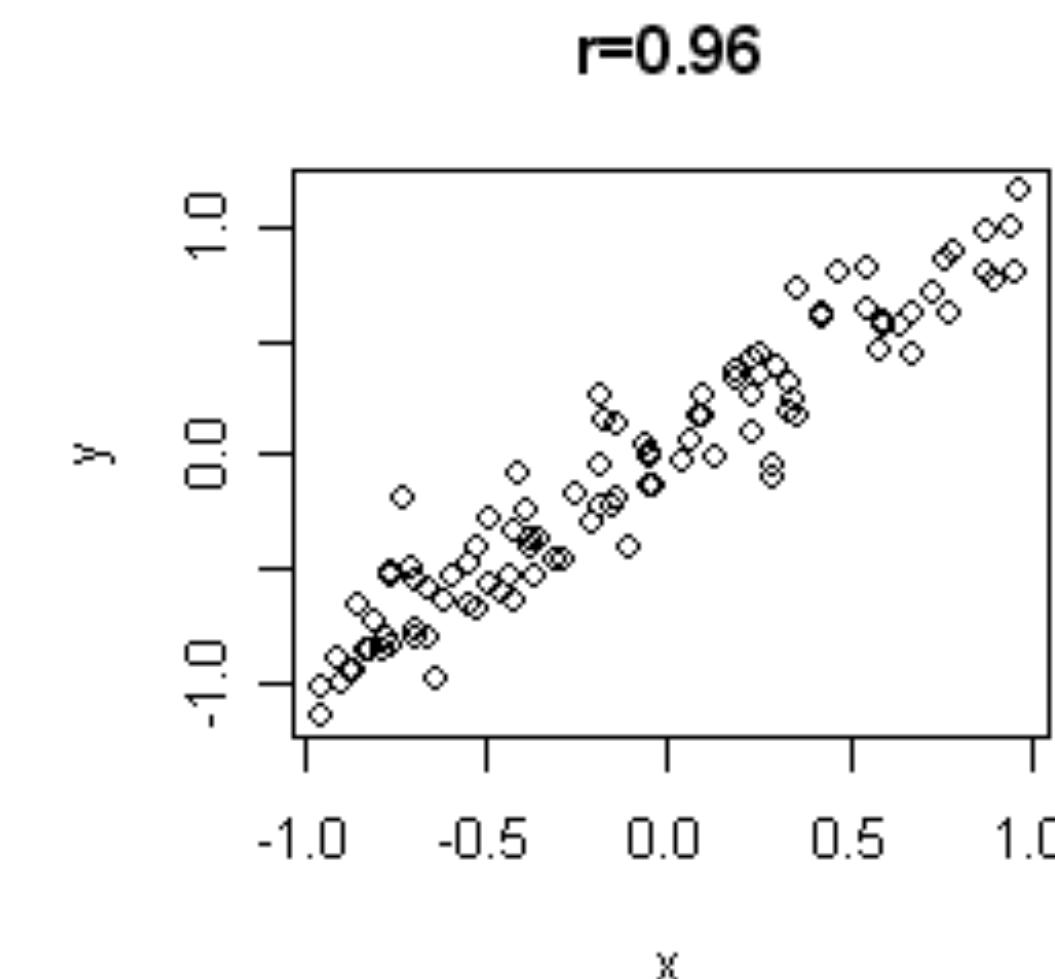
- Great for comparing different sized samples
- Nobody understands them

# Visualizing Distributions: Others

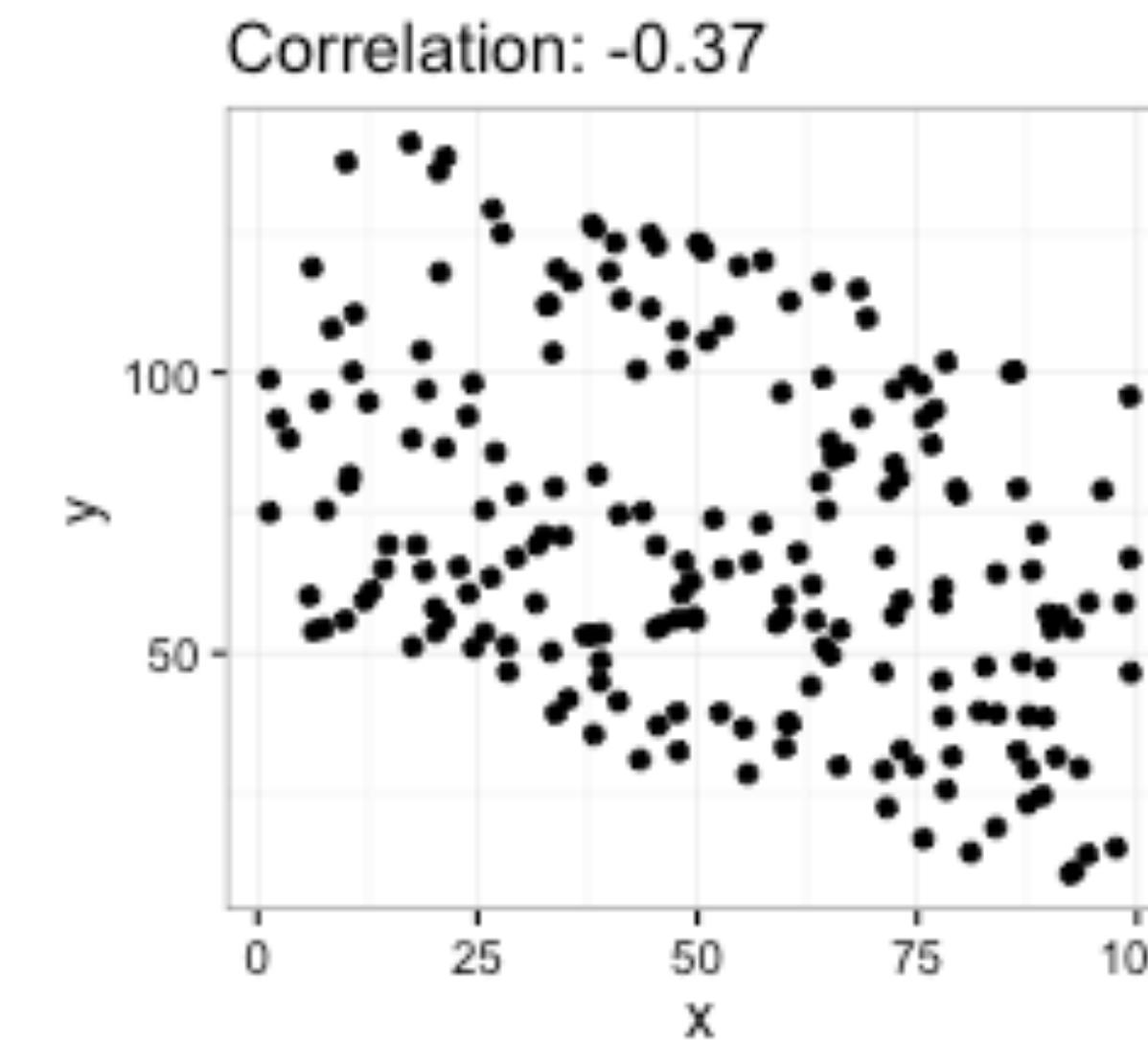
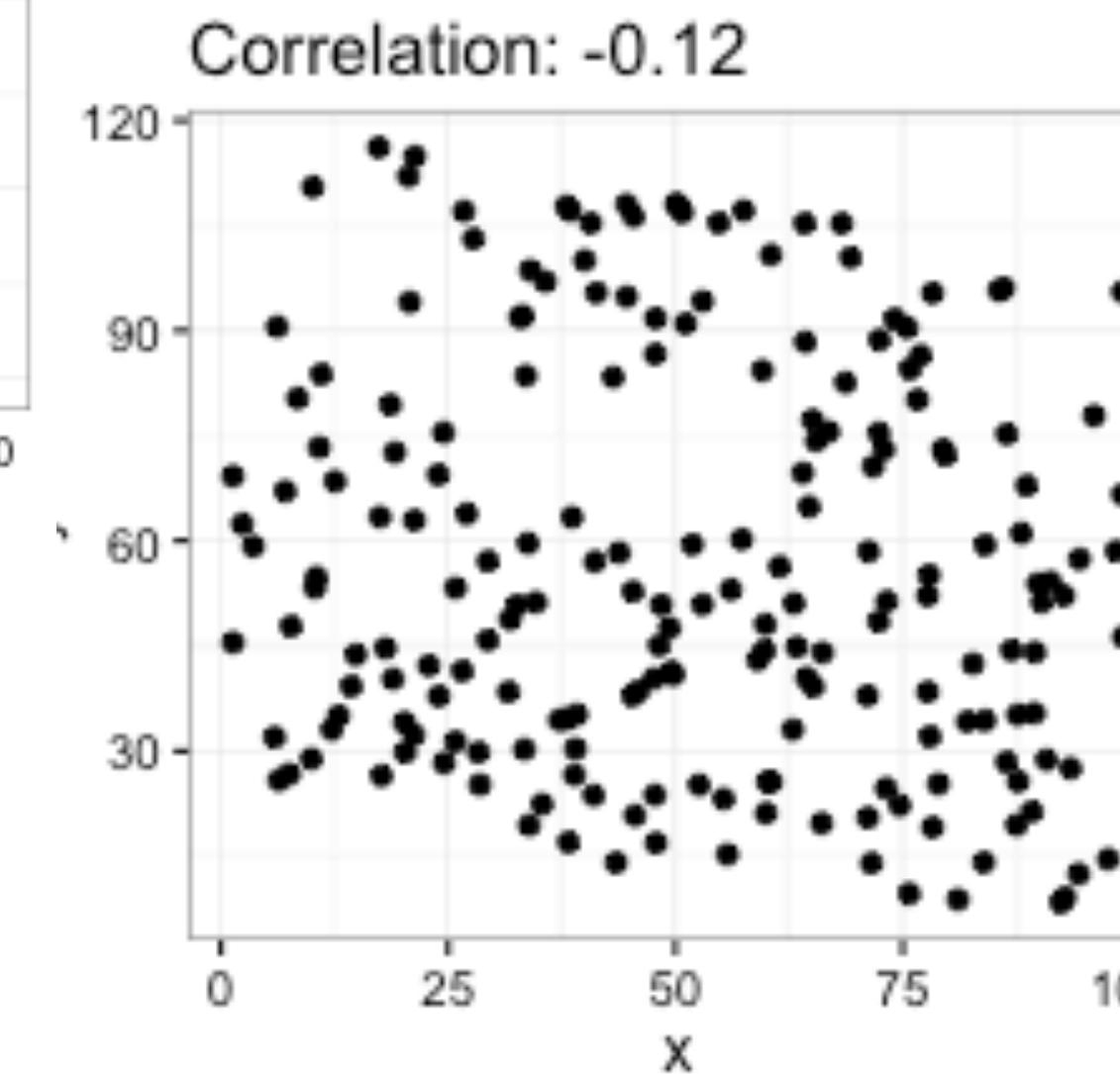
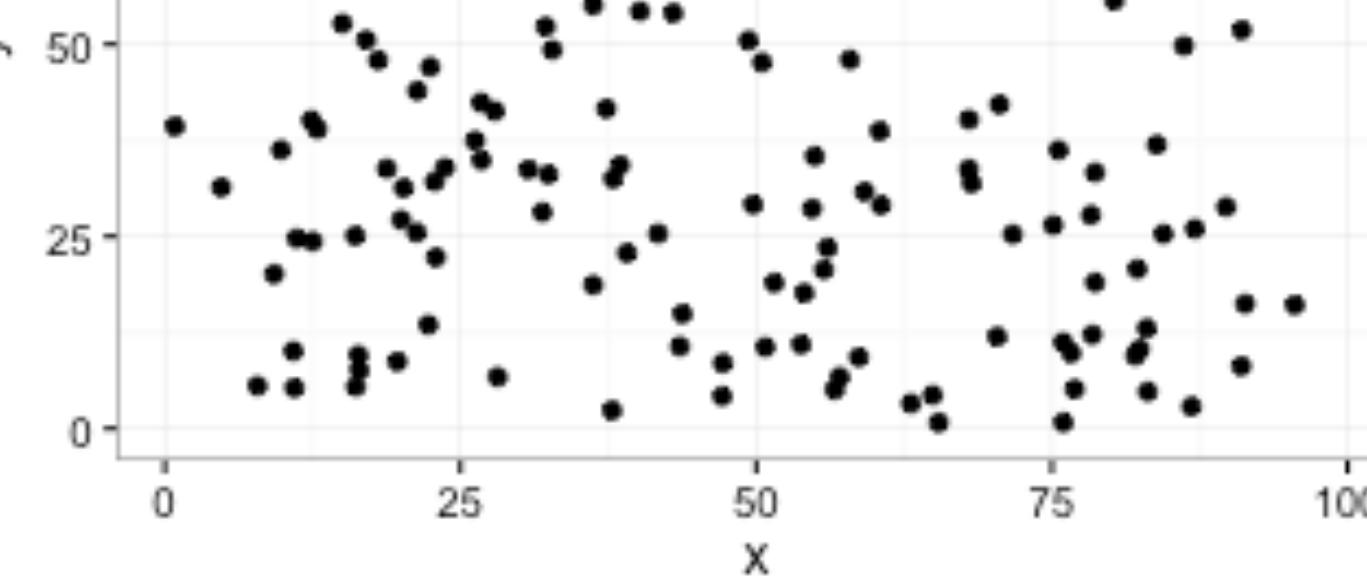
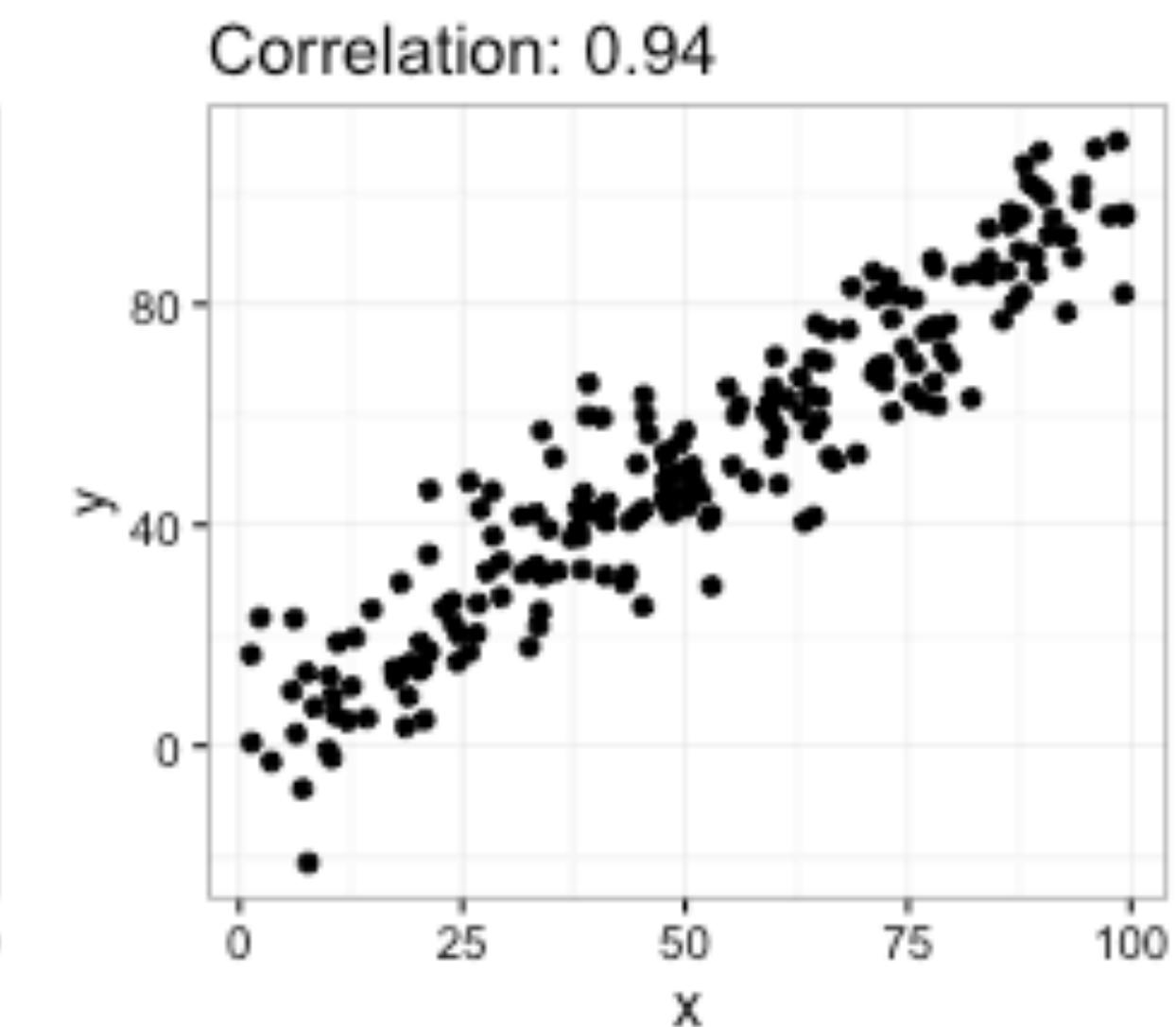
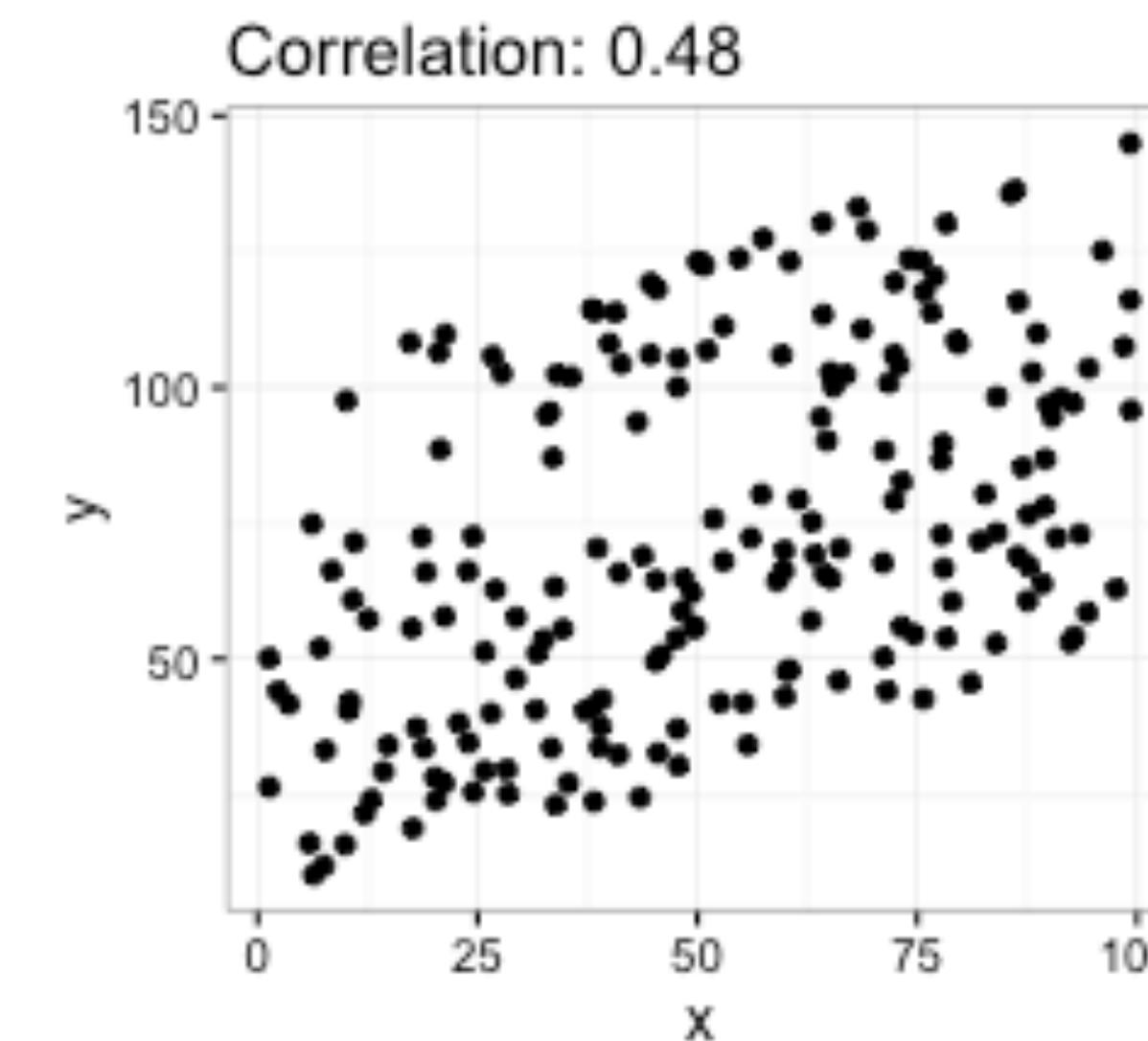
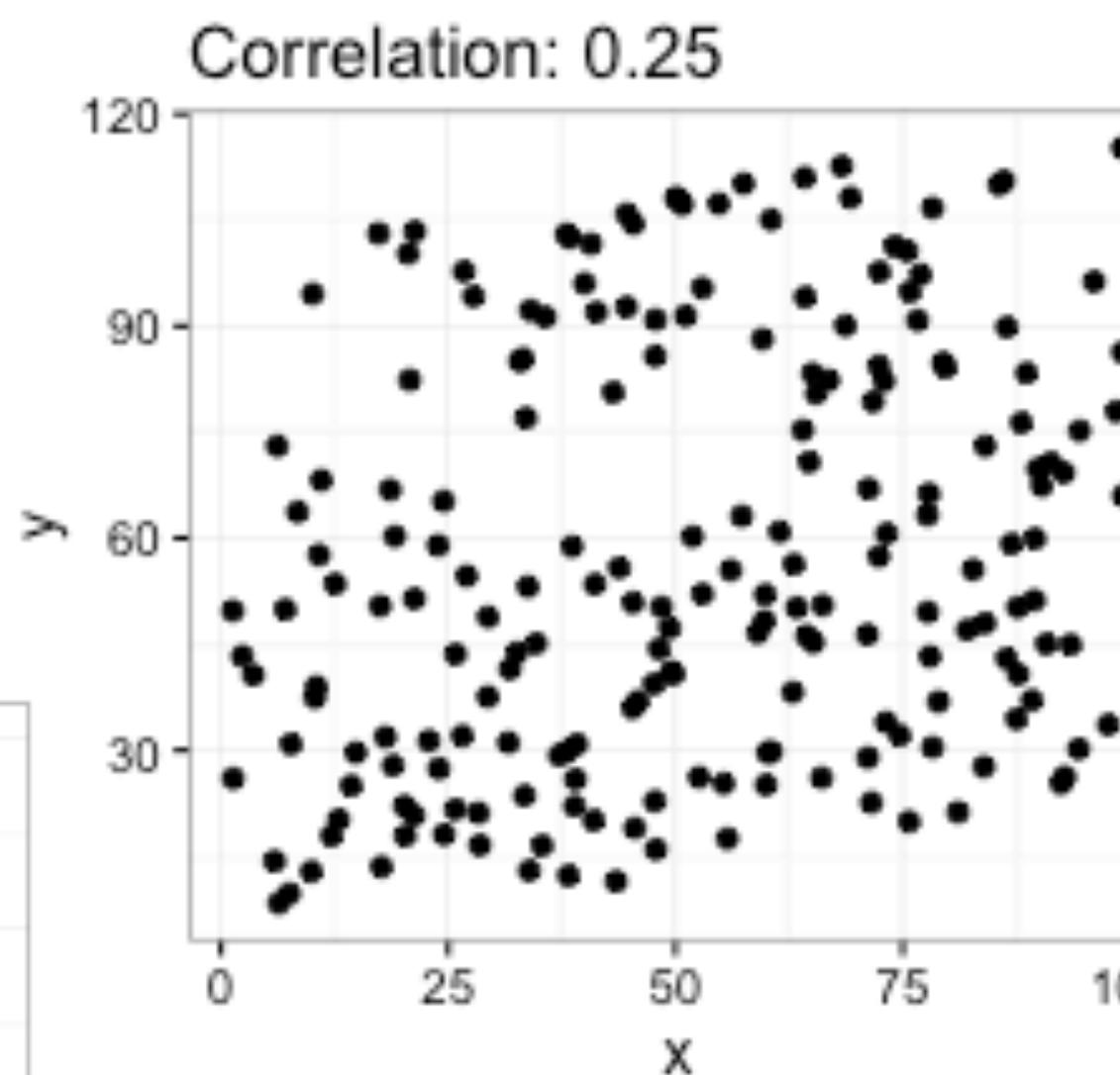
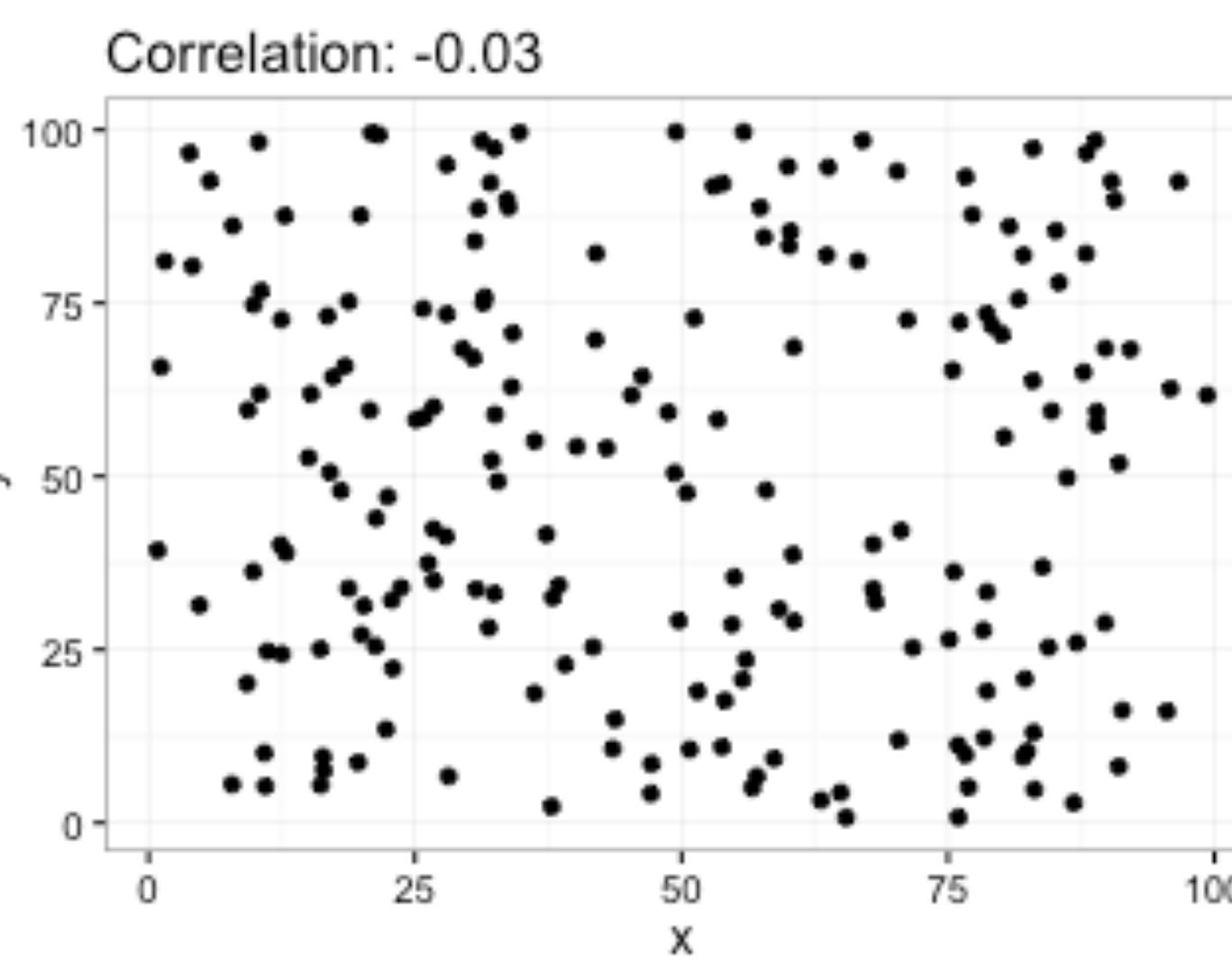


# Correlations

- 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



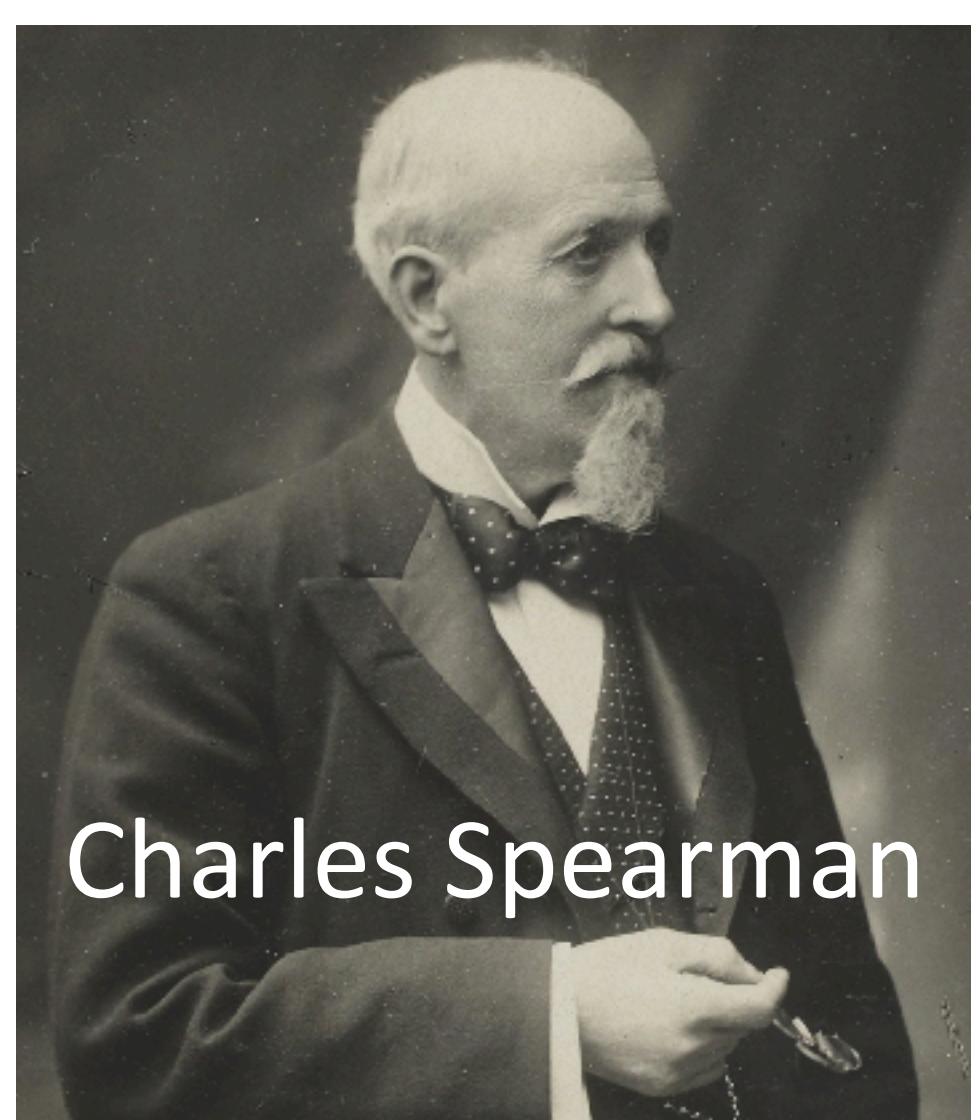
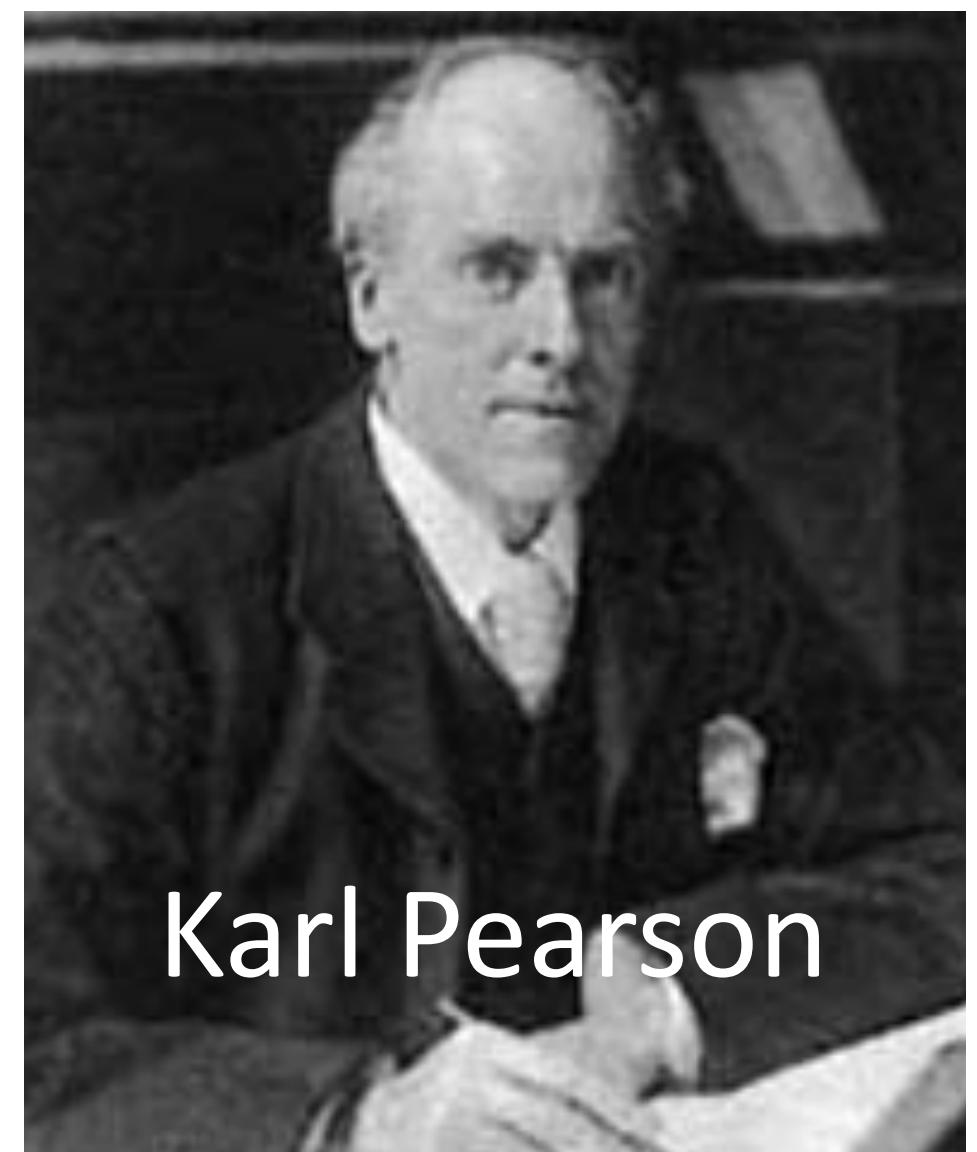
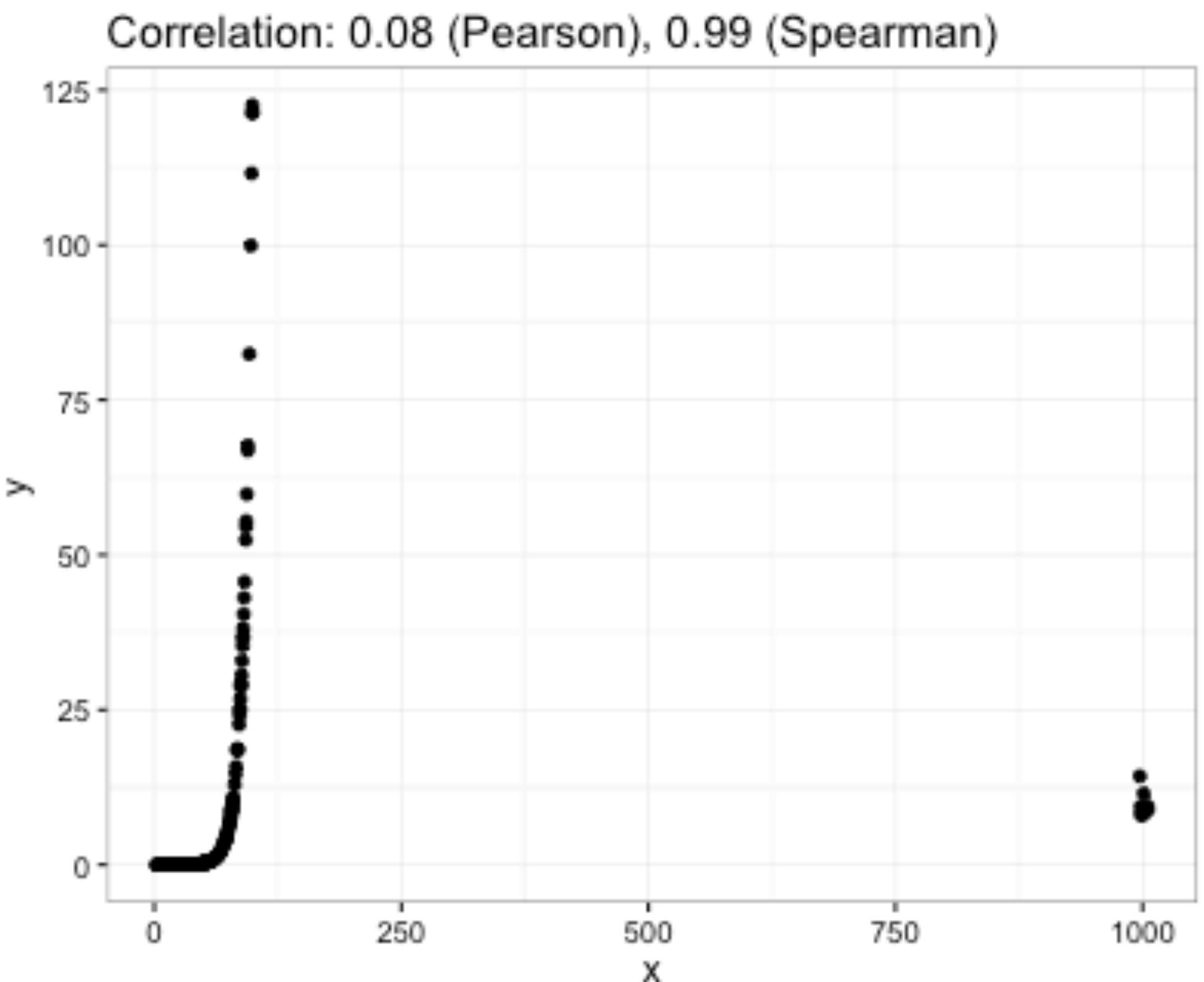
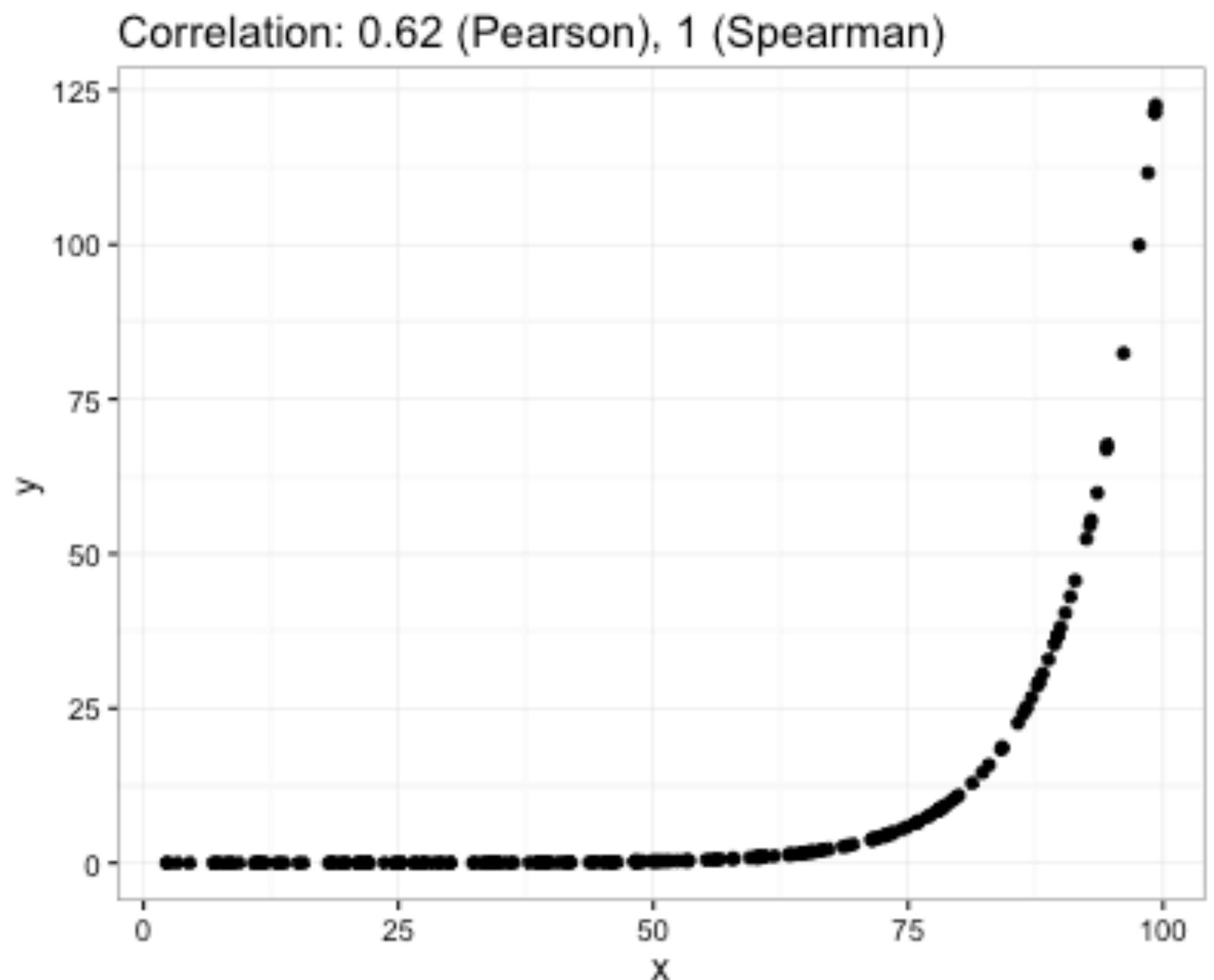
# Correlations



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



## confidence interval

/'kænfəd(ə)ns 'in(t)ərvəl/

*noun*

STATISTICS

### Wrong (intuitive) definition:

“A range of values so defined that there is a specified probability that the value of a parameter lies within it.”

### Correct (confusing) definition:

“If the same population is sampled on numerous occasions and interval estimates are made on each occasion, the resulting intervals would bracket the true population parameter in approximately 95 % of the cases.”

## confidence interval

/'kænfəd(ə)ns 'in(t)ərvəl/

*noun* STATISTICS

- Probability of flipping a heads with a coin?
- What if we get 4 out of 10 coin flips?
- What if we get 400 out of 1000 coin flips?

What's the difference in the last two?

# Confidence Intervals

**4 out of 10**

$$p = \frac{4}{10} = 0.4$$

$$CI = 0.4 \pm 1.96 \sqrt{\frac{0.4(1-0.4)}{10}}$$

$$CI = 0.4 \pm 0.3036$$

$$CI = p \pm z_{1-\alpha/2} \sqrt{\frac{p(1-p)}{n}}$$

where  $p$  = proportion of interest

$n$  = sample size

$\alpha$  = desired confidence

$z_{1-\alpha/2}$  = “z value” for desired level of confidence

$z_{1-\alpha/2} = 1.96$  for 95% confidence

$z_{1-\alpha/2} = 2.57$  for 99% confidence

$z_{1-\alpha/2} = 3$  for 99.73% confidence

**95% CI:**

**9.6% and 70.4%**

**400 out of 1000**

$$p = \frac{400}{1000} = 0.4$$

$$CI = 0.4 \pm 1.96 \sqrt{\frac{0.4(1-0.4)}{1000}}$$

$$CI = 0.4 \pm 0.0303$$

**95% CI:**

**37% and 43%**

# 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?*

	<b>Severe</b>	<b>Incidents</b>	<b>%</b>
<b>Last year</b>	21	88	23.8%
<b>This year</b>	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?*

	<b>Severe</b>	<b>Incidents</b>	<b>%</b>	<b>95% lower</b>	<b>95% upper</b>
<b>Last year</b>	21	88	23.8%	15.4%	34.1%
<b>This year</b>	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

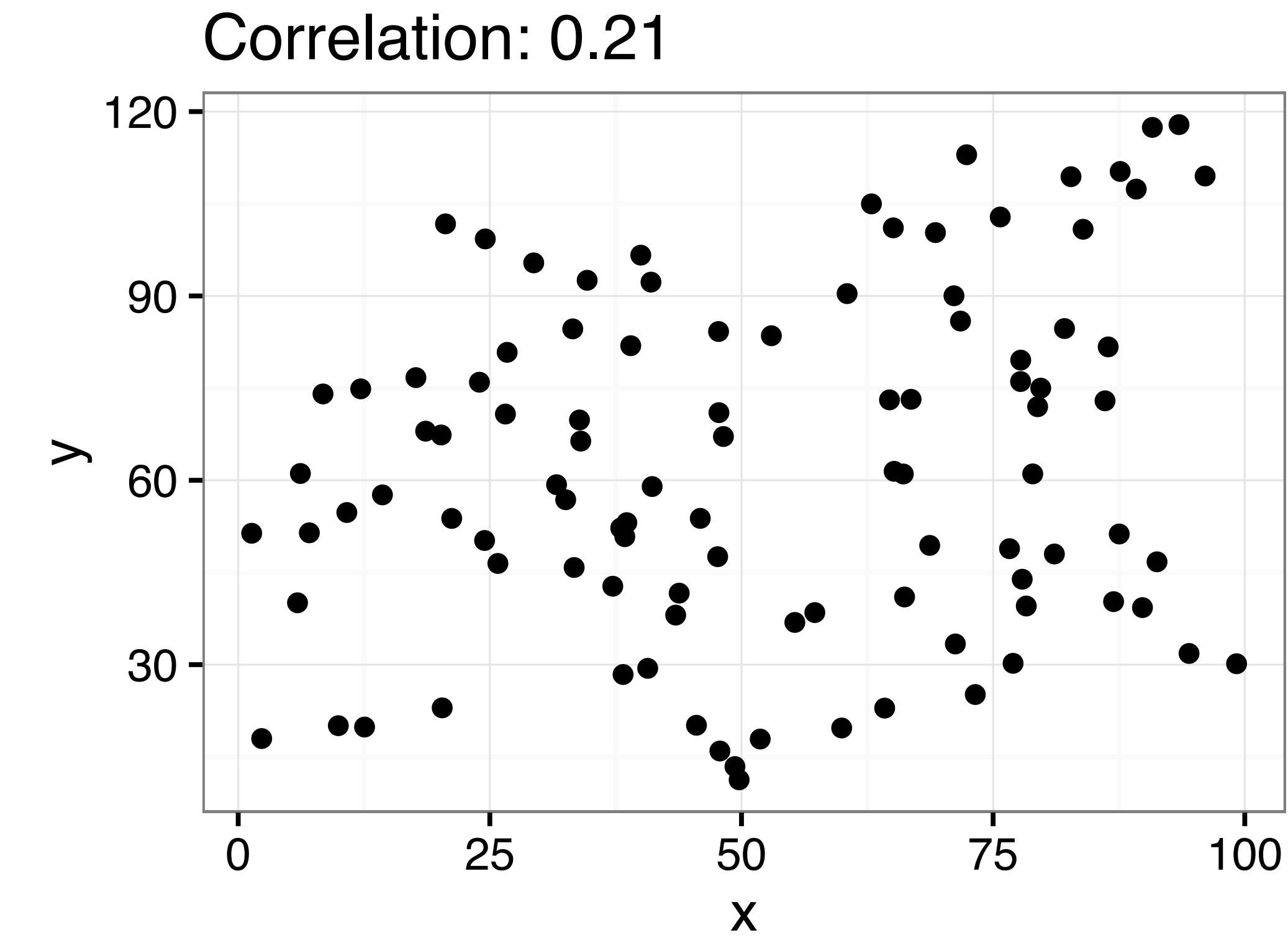
- With 100 values of each  $x$  and  $y$ , we estimate a correlation of 0.21
- What is the 95% confidence around “true” correlation?

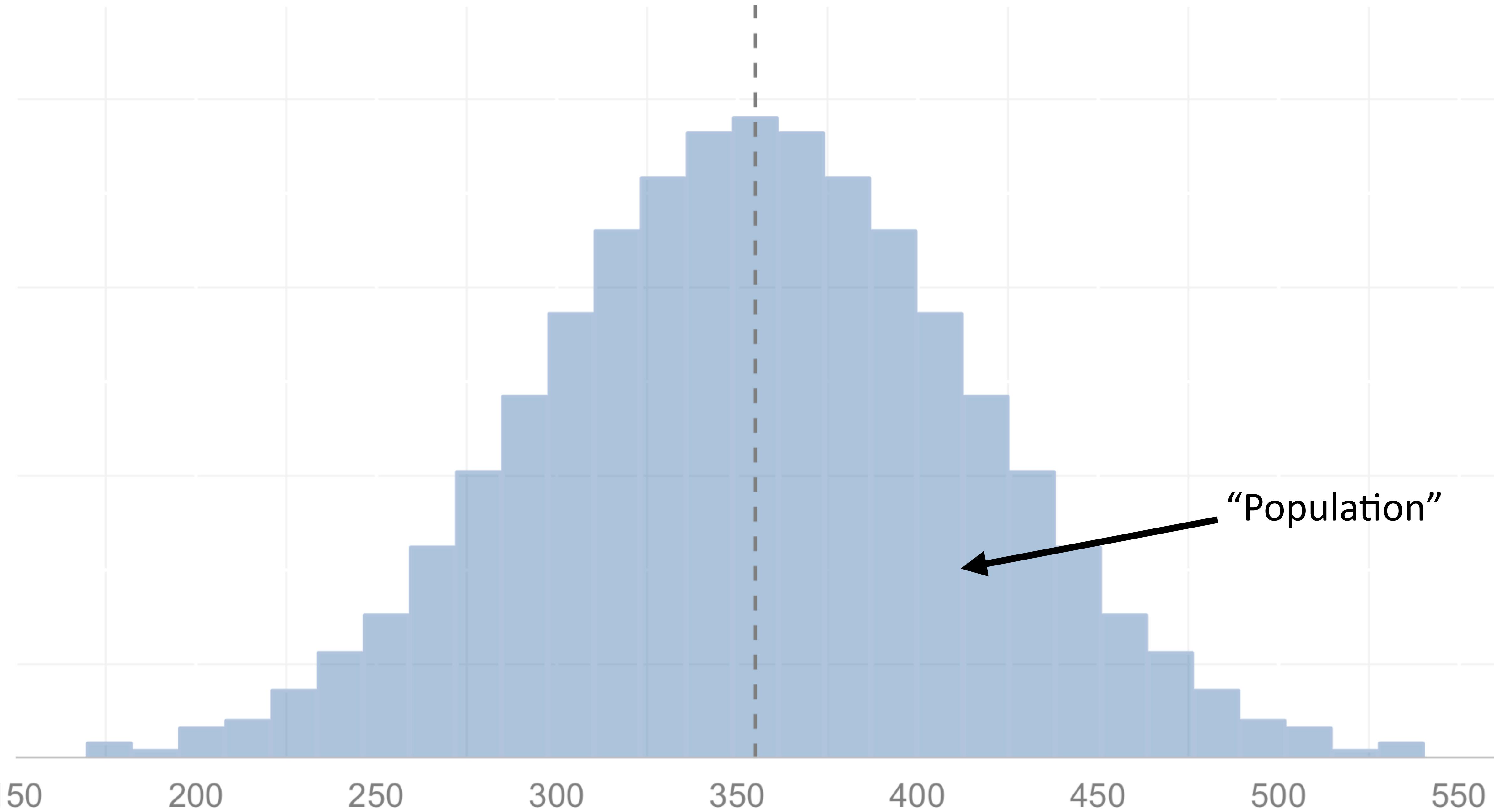
$r$  is between 0.01 and 0.39

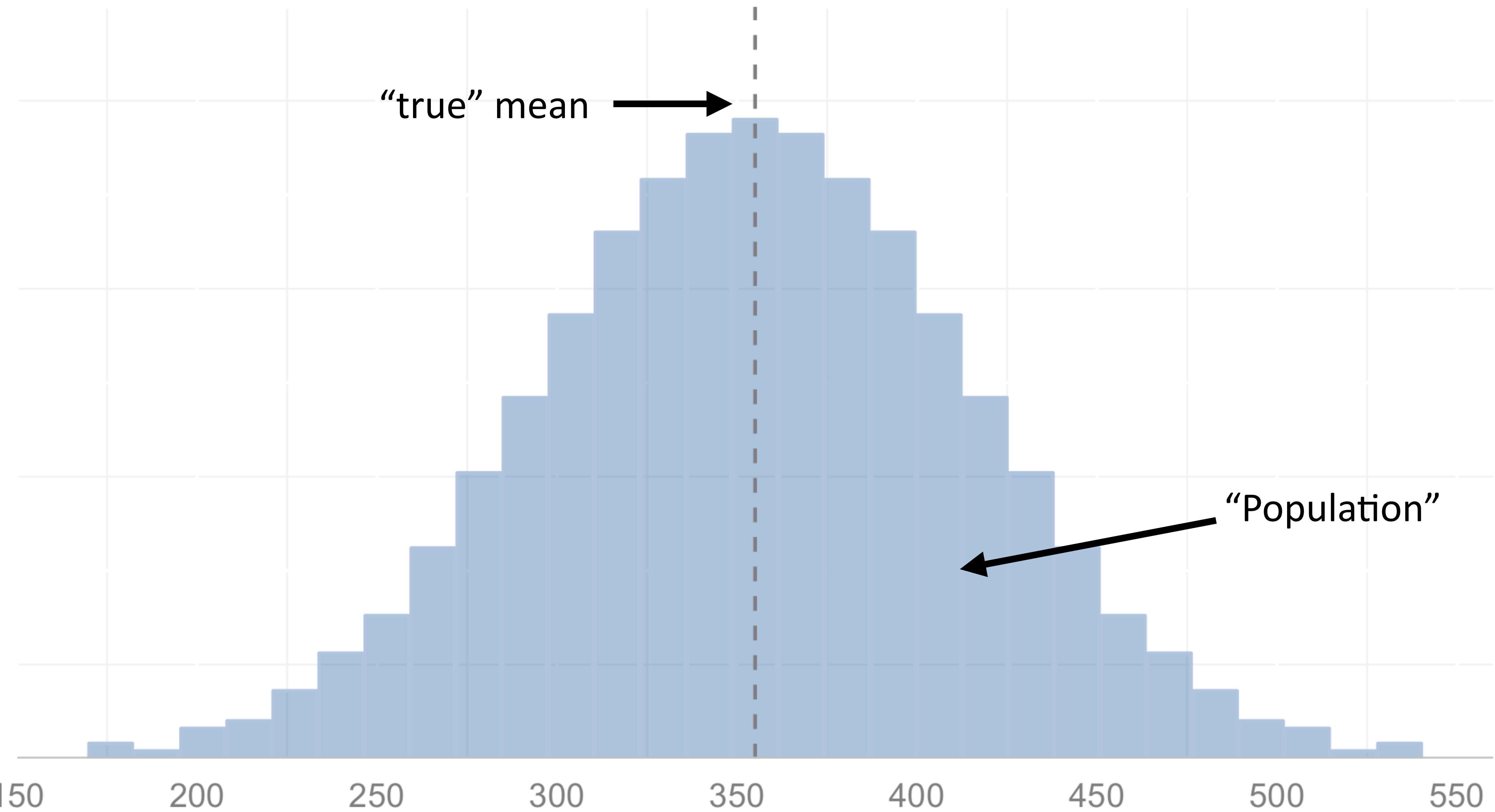
```
> cor.test(x,y)  
... (hypothesis testing previously shown)
```

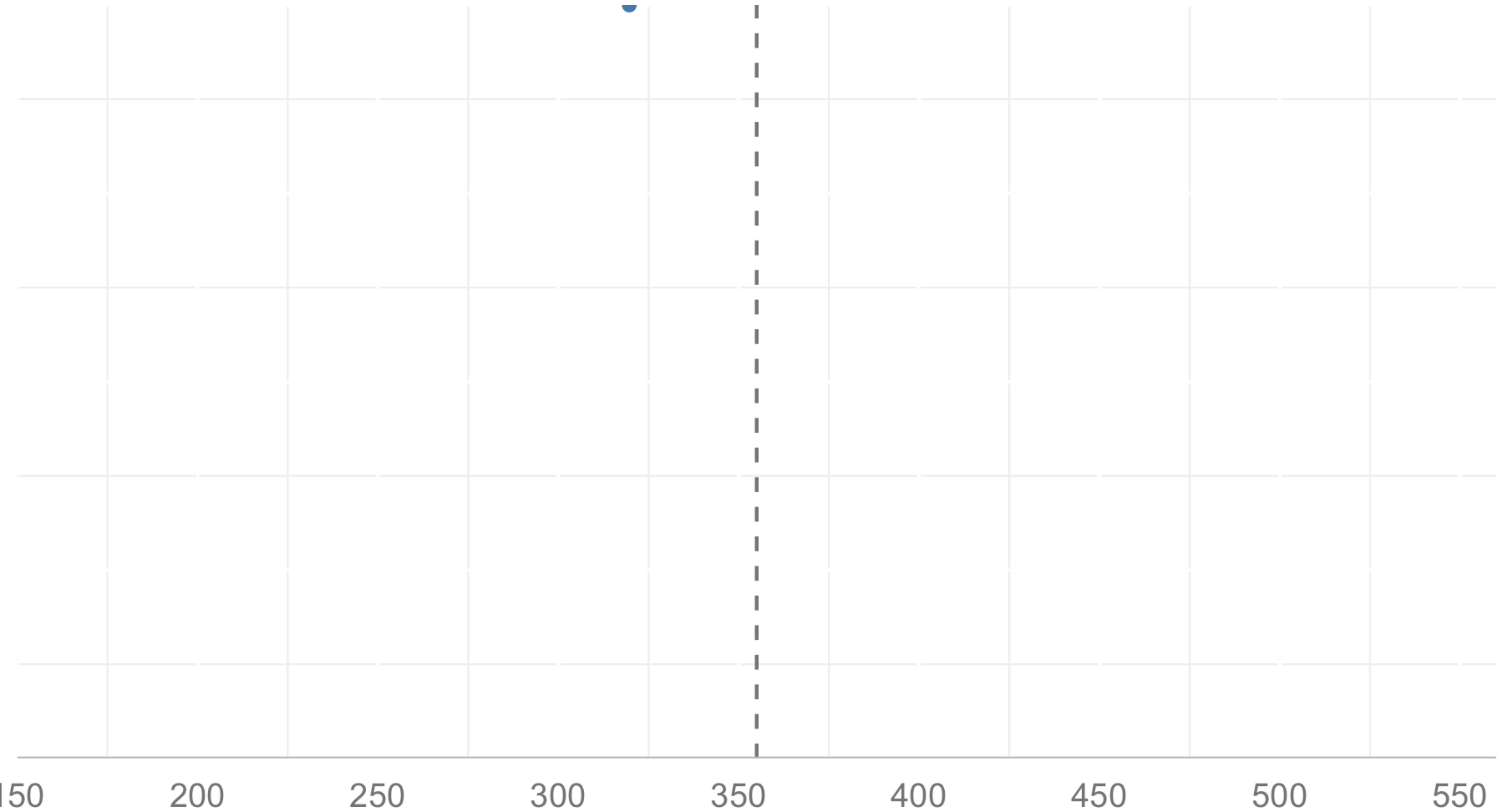
95 percent confidence interval:

0.01334506 0.38962194





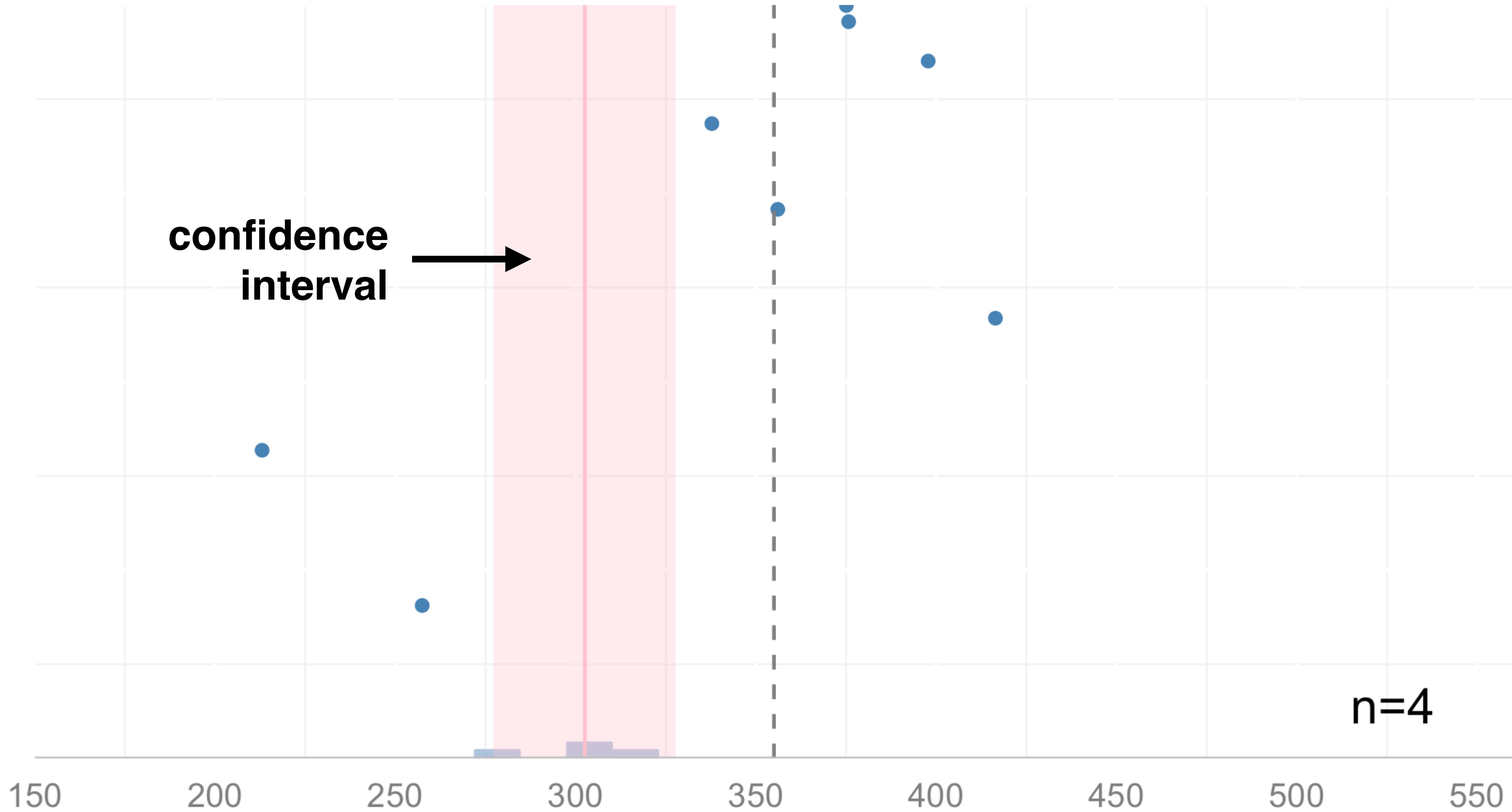




**confidence  
interval**

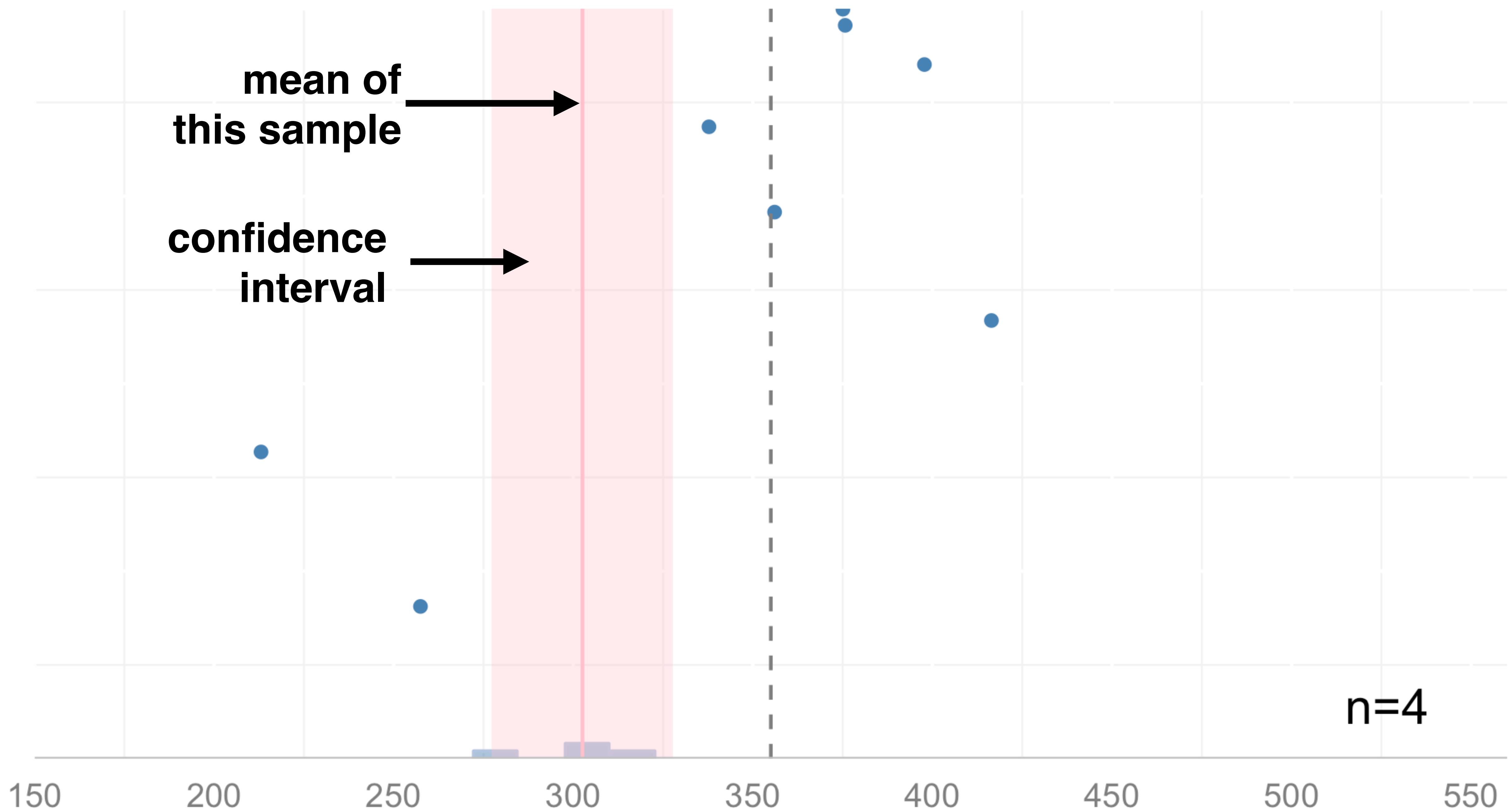


$n=4$

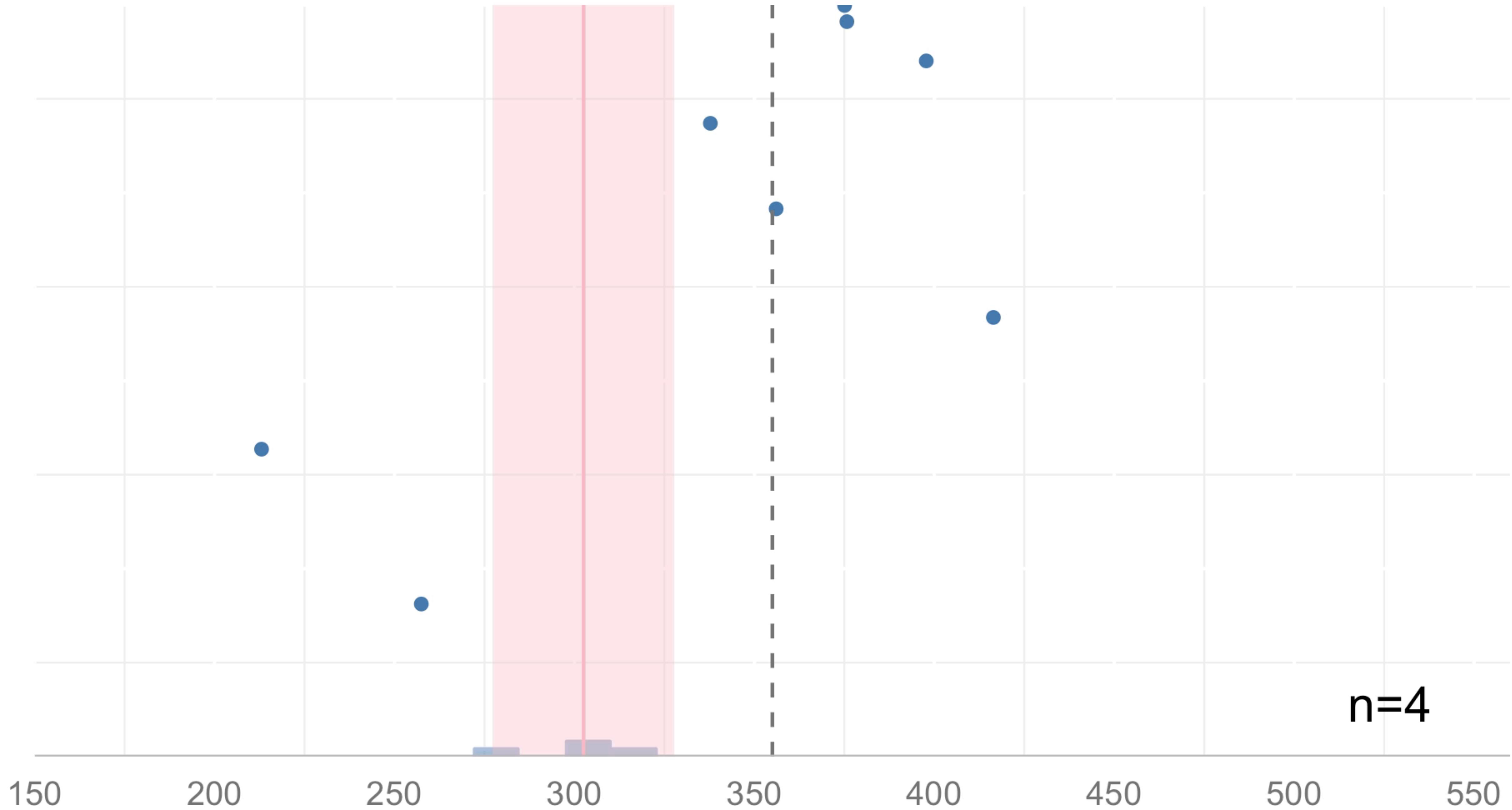


**mean of  
this sample**

**confidence  
interval**

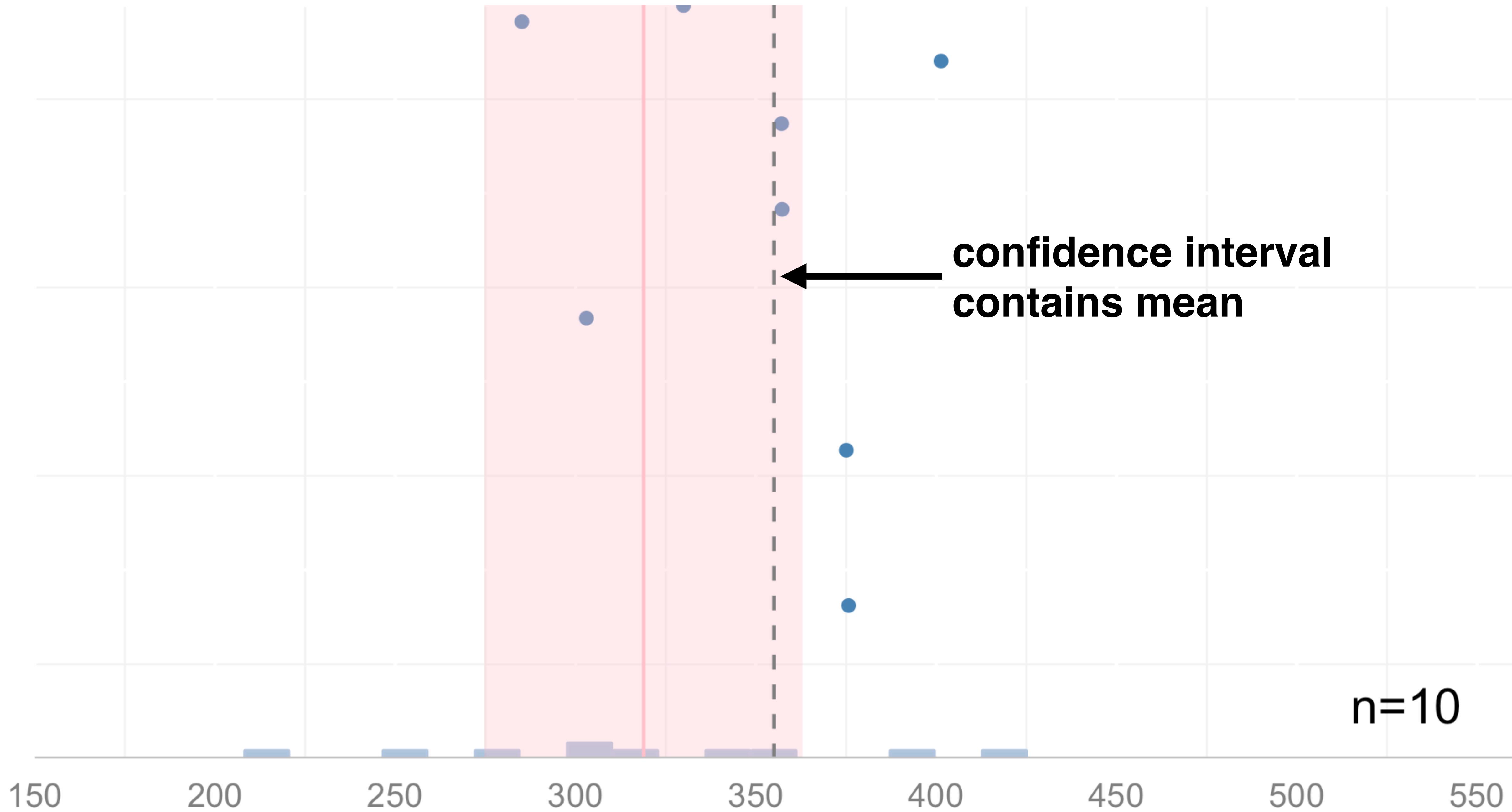


$n=4$

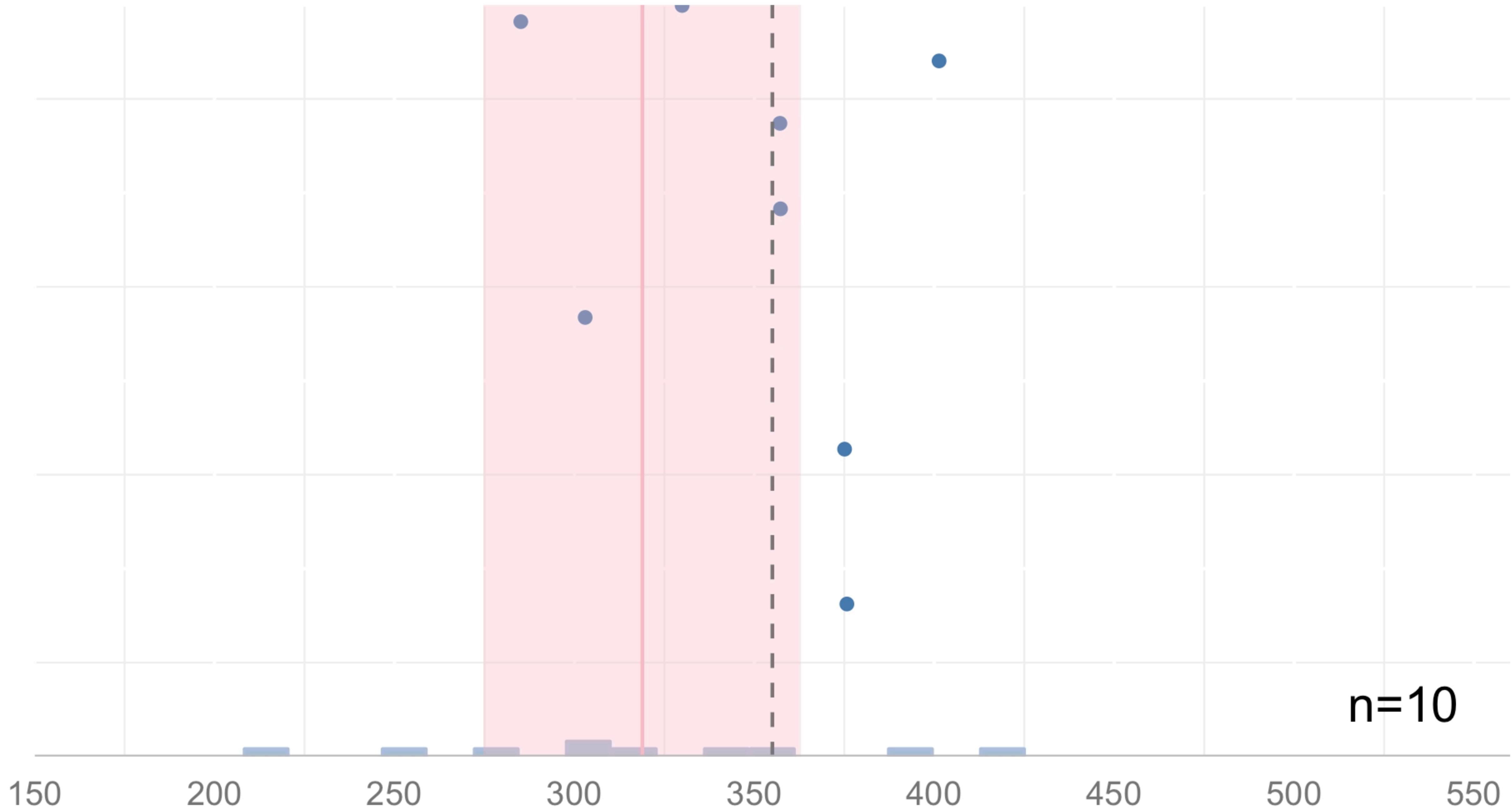


$n=10$

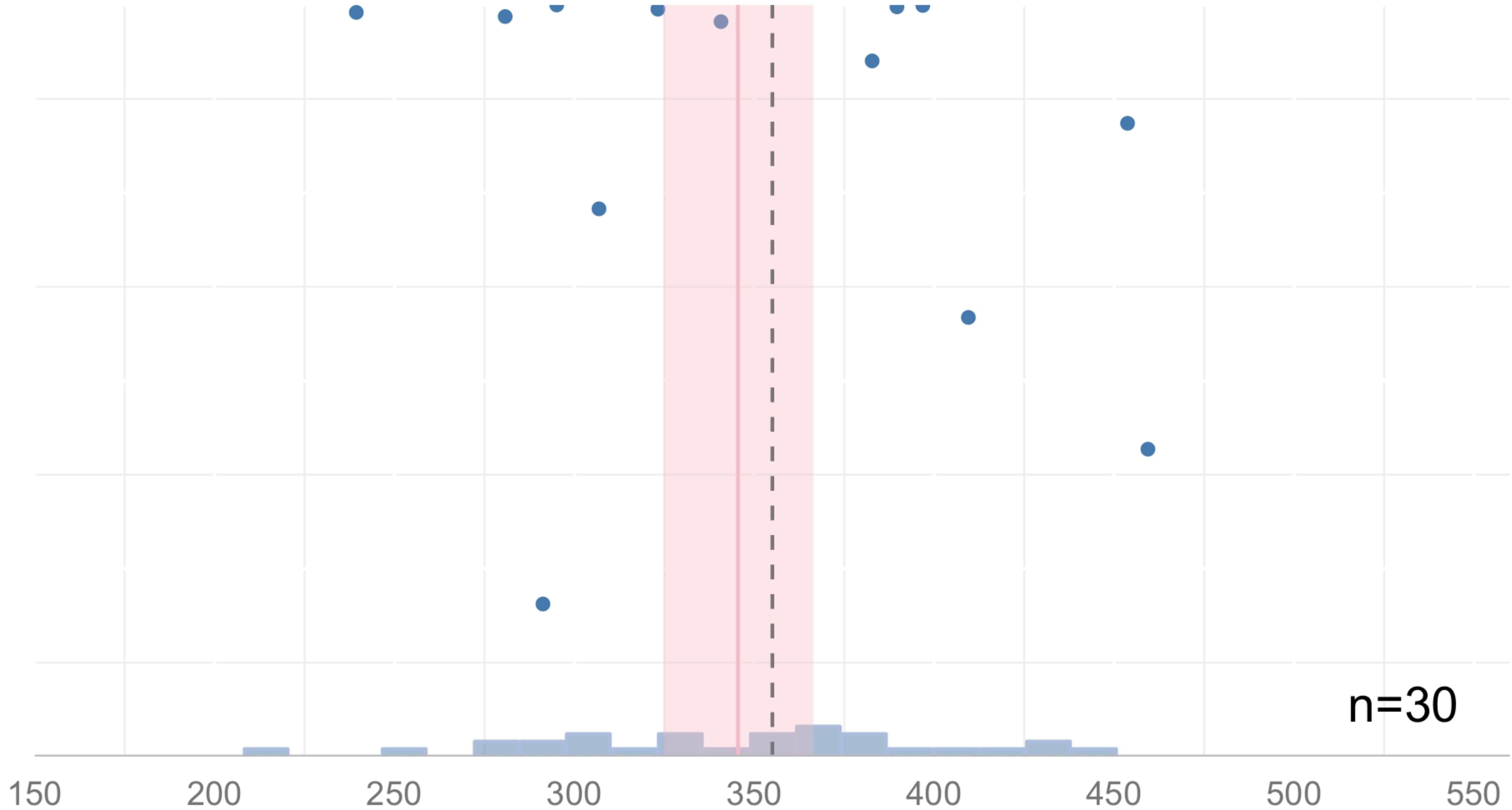
**confidence interval  
contains mean**

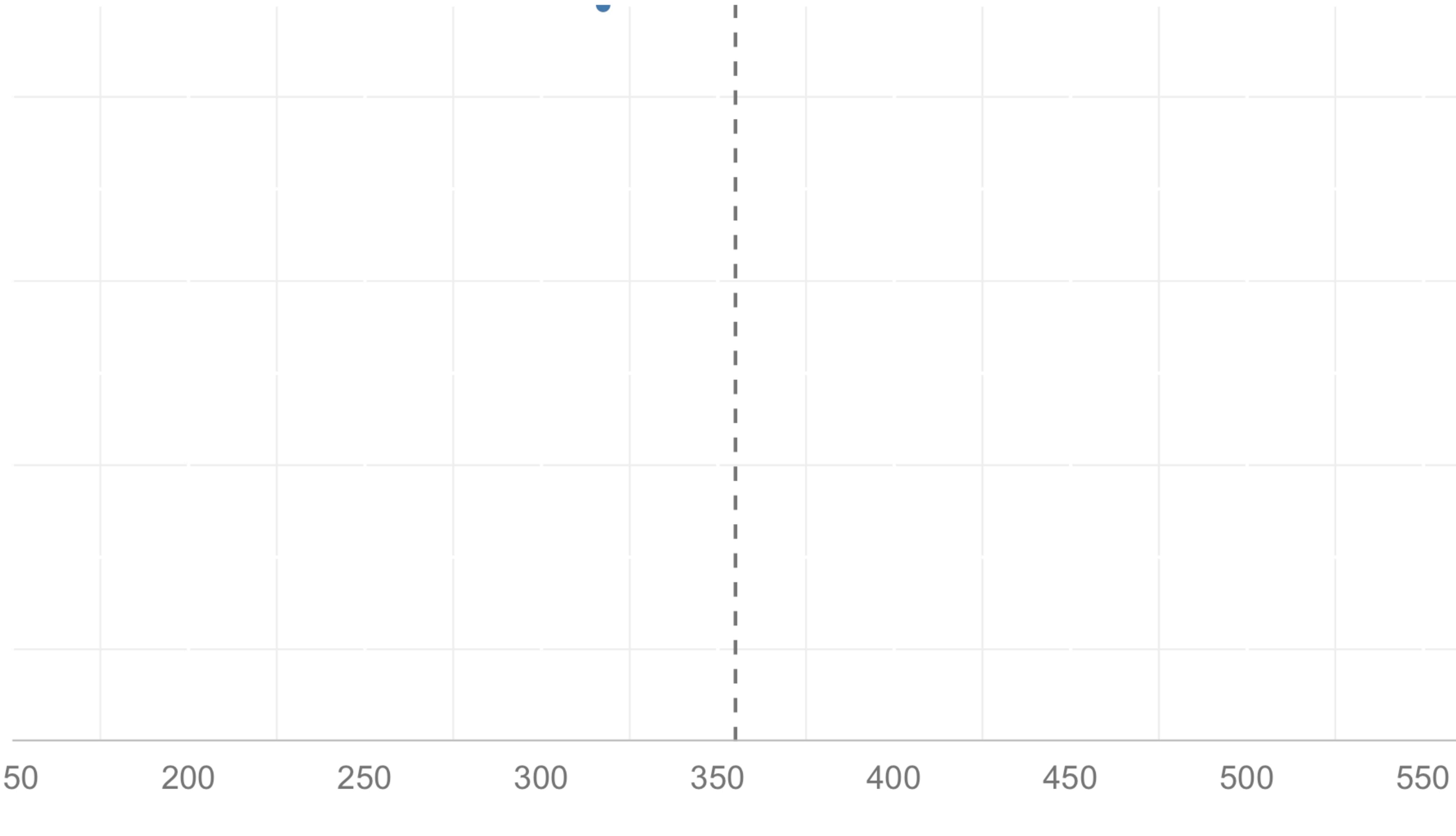


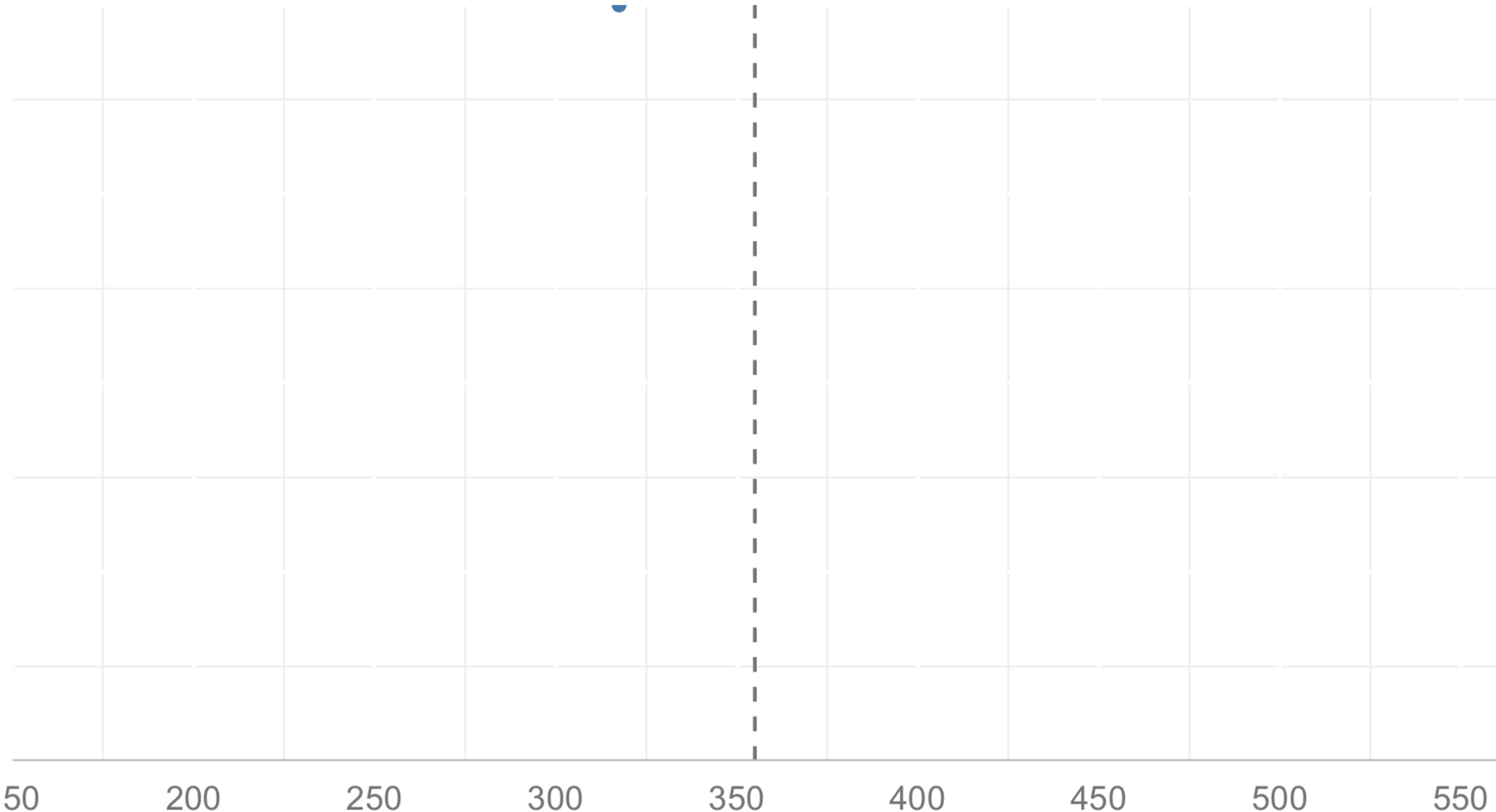
$n=10$

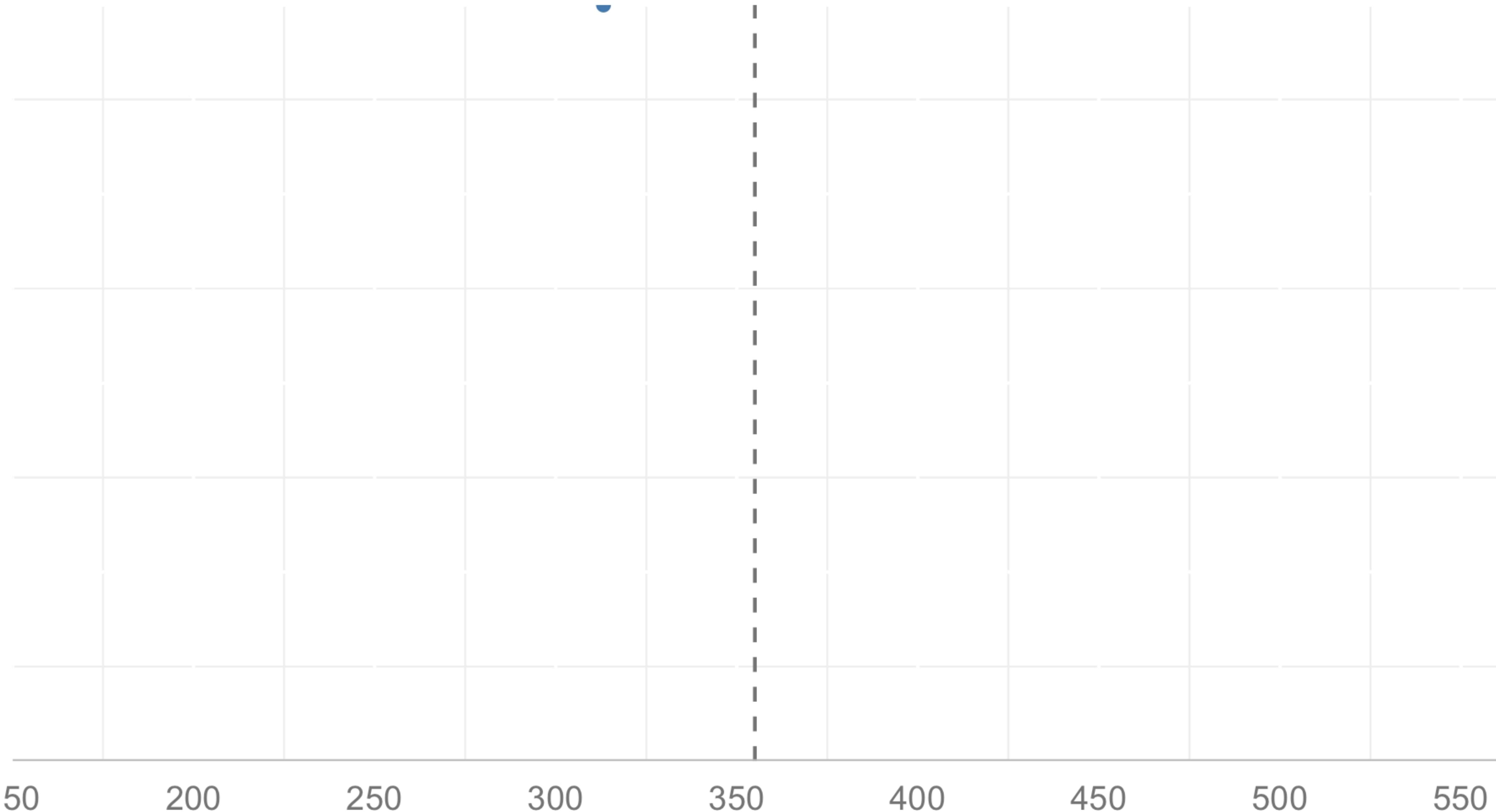


$n=30$









# Hypothesis Testing: p-value

A “p-value” is the probability of obtaining an effect at least as extreme as the one in your sample data, assuming the truth of the null hypothesis.

**Example:** We are conducting an internal phishing campaign. We want to explore if there is a noticeable difference between the HR department and the marketing department.

Null Hypothesis ( $H_0$ ): There is no difference ( $HR == \text{Marketing}$ )

Alt. Hypothesis ( $H_a$ ): There is a difference ( $HR != \text{Marketing}$ )

# Hypothesis Testing: p-value

**Example:** We are conducting an internal phishing campaign. We want to explore if there is a noticeable difference between the HR department and the marketing department.

10 of the 20 HR employees (50%) clicked on internal phishing emails.

6 of the 30 marketing employees (20%) clicked on internal phishing emails.

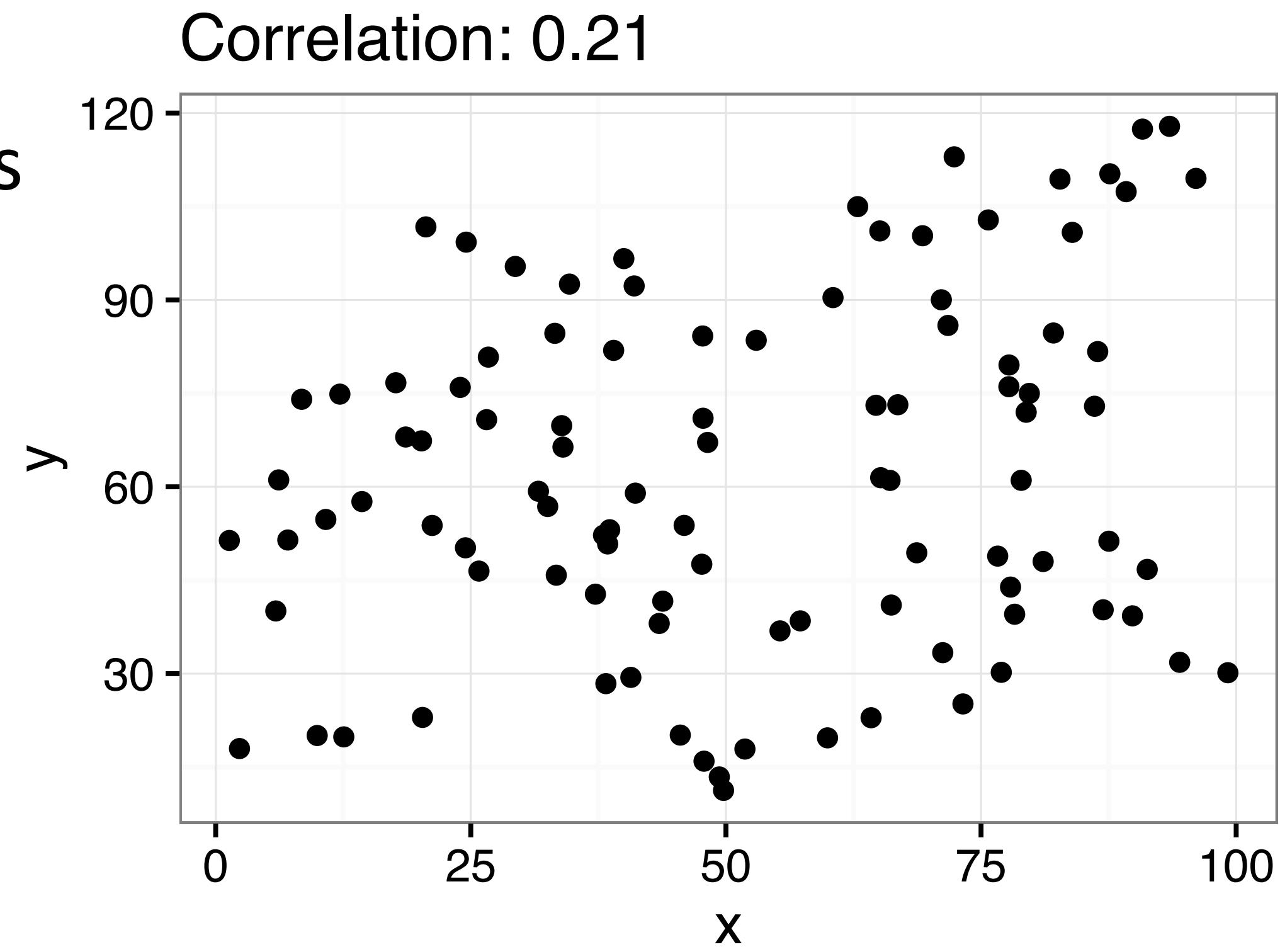
		Clicked?	
		No	Yes
Dept	HR	10	10
	Marketing	24	6

two-proportions z-test:  
p-value = 0.05506

We cannot reject the null hypothesis at the 0.05 significance level. (the sample sizes are just too small to detect a noticeable difference)

# 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?
  - $p\text{-value} = 0.03671$
- 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)
```

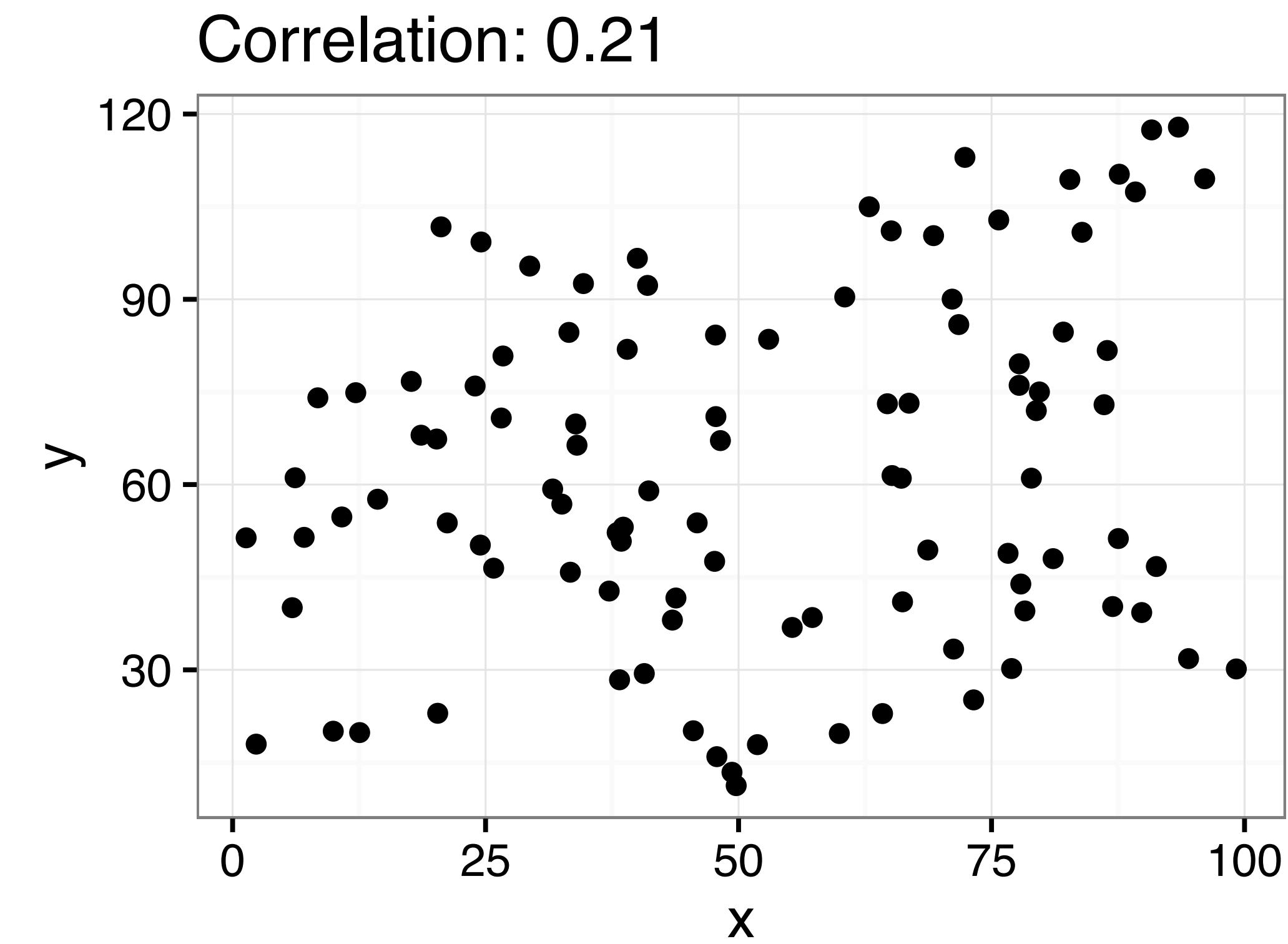
Pearson's product-moment correlation

data: x and y

t = 2.118, df = 98, p-value = 0.03671

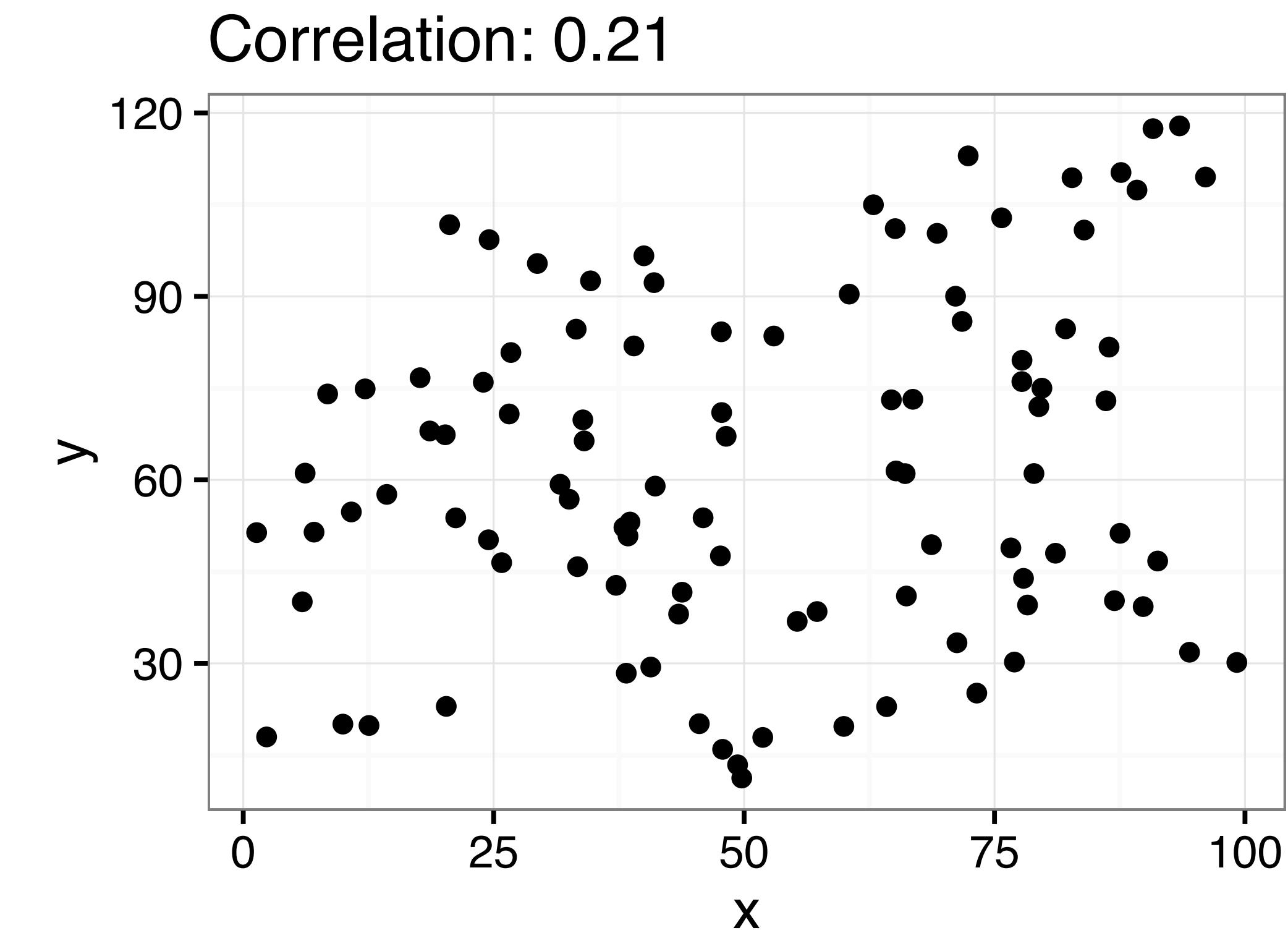
alternative hypothesis: true correlation is not equal to 0

... (more output coming up)



# Correlation p-value in python

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



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

O'REILLY®

# Security

BUILD BETTER DEFENSES

[oreillysecuritycon.com](http://oreillysecuritycon.com)

#oreillysecurity

**EDA in code**

# Setting up for visualization

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

Python

```
library(tidyverse) # will load ggplot2  
  
# OR call them directly  
library(ggplot2)
```

R

# 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)

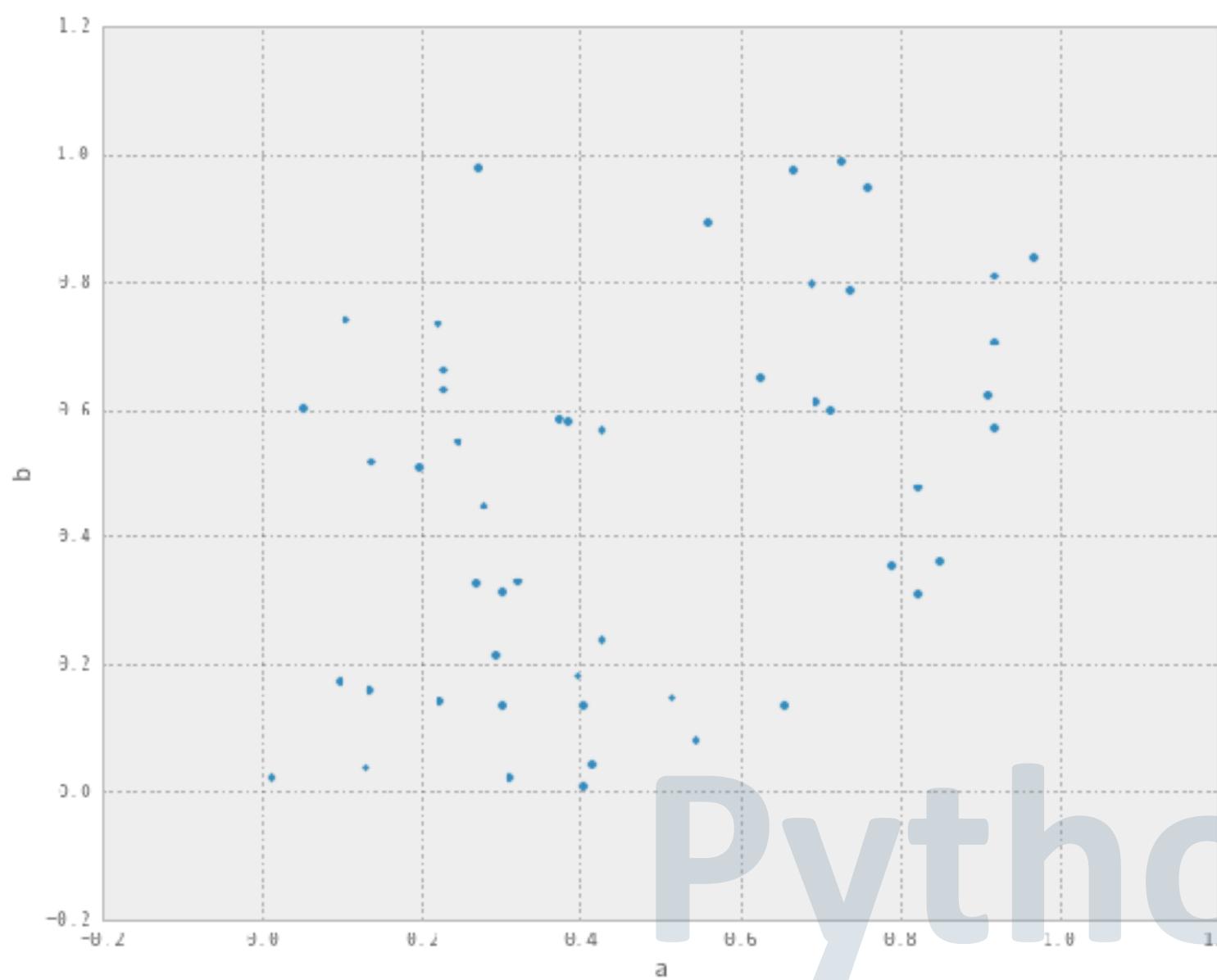
# ggplot2

- 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

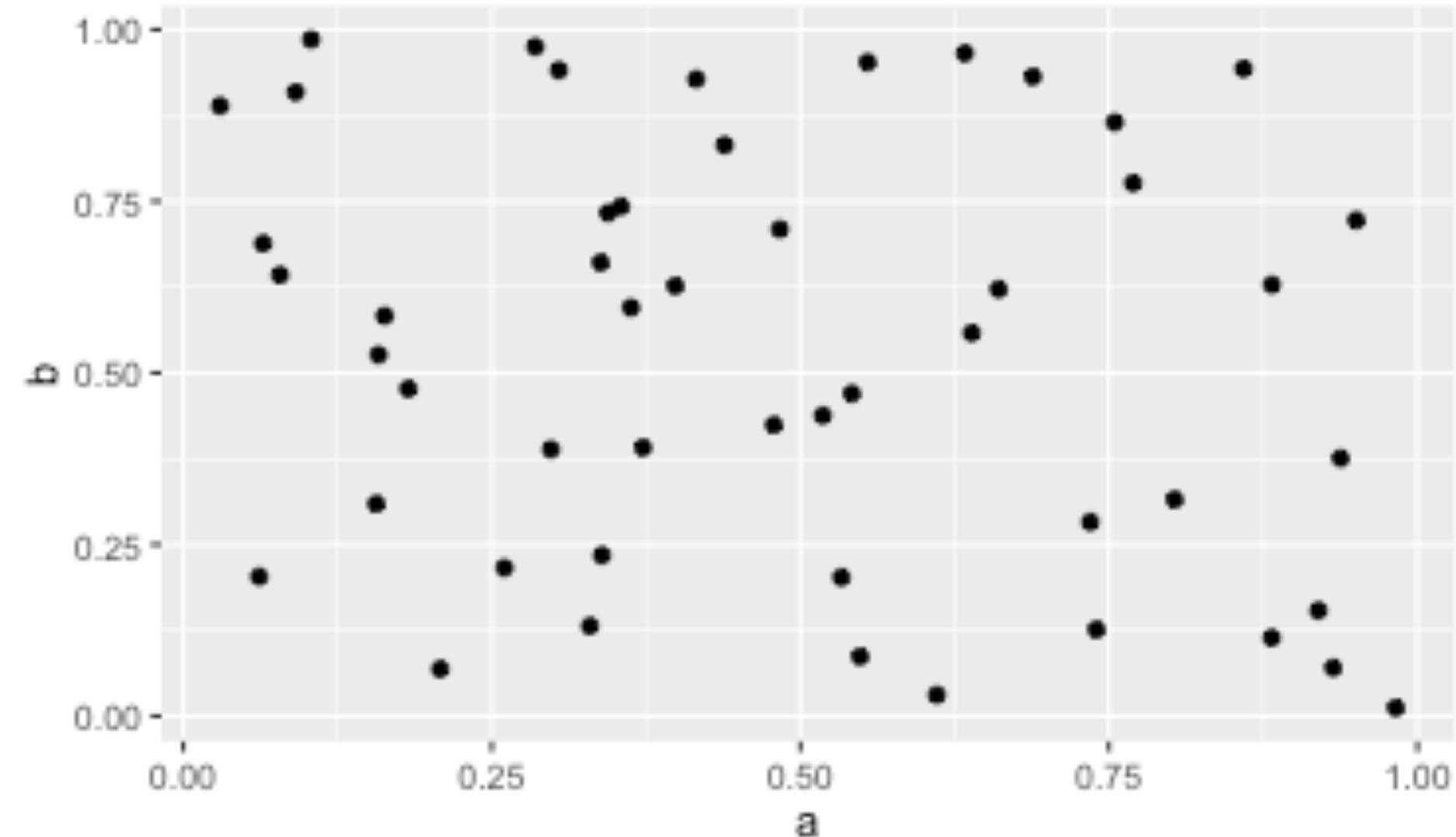
# Scatter Plot

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



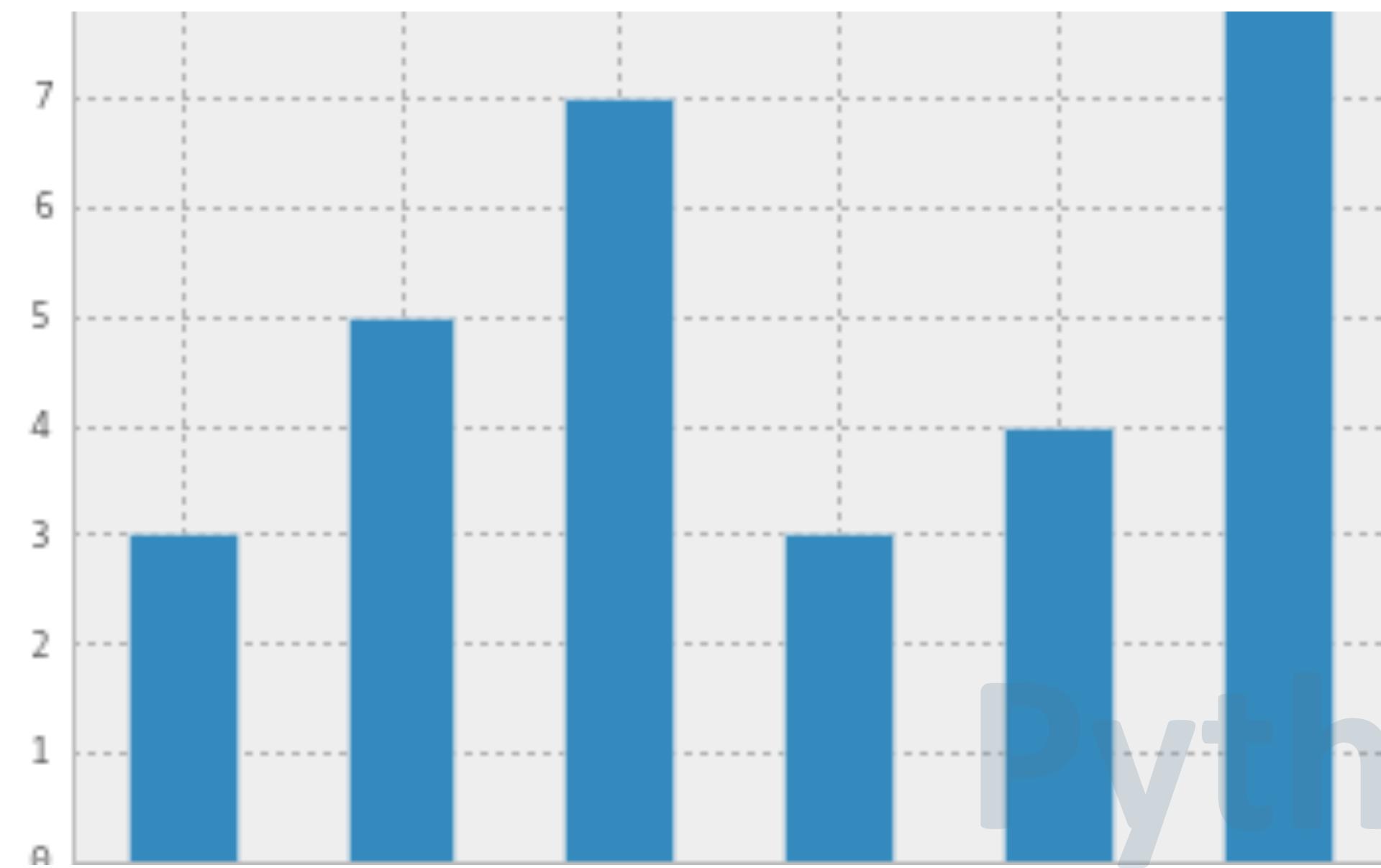
Python

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

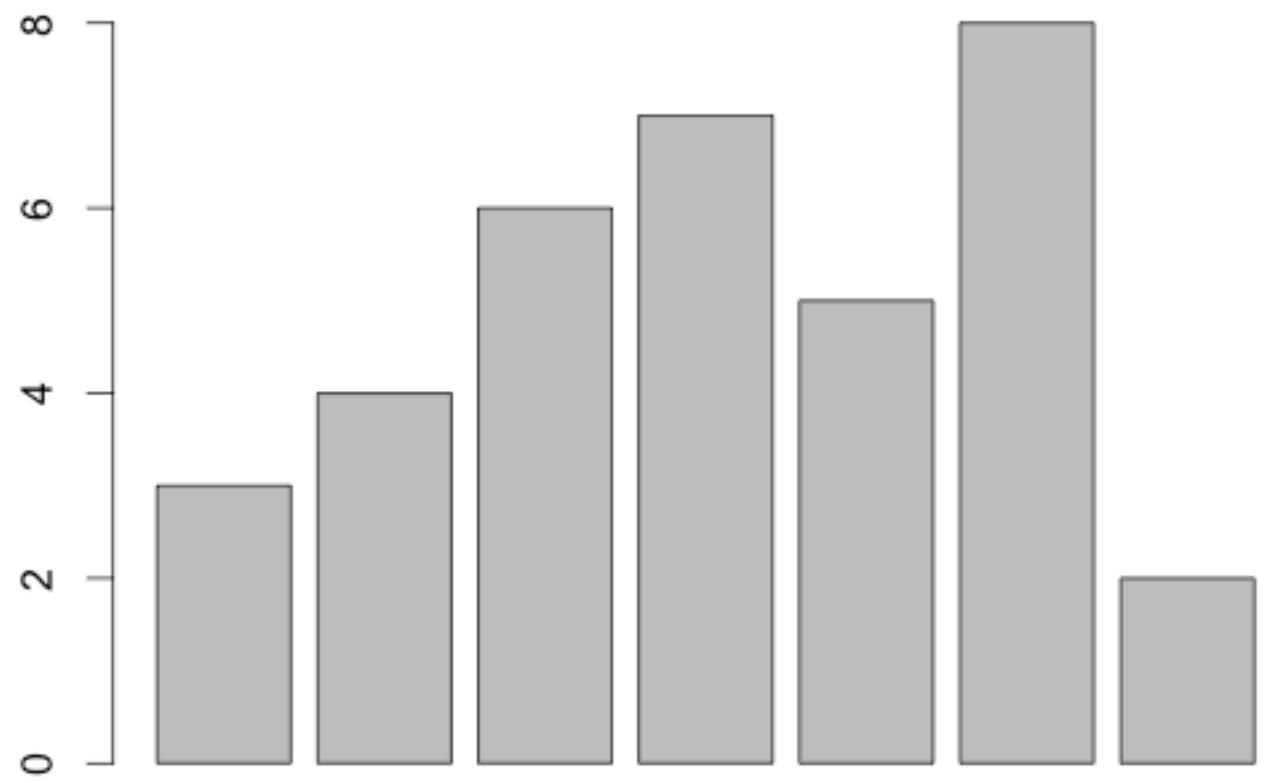


# Setting up for visualization

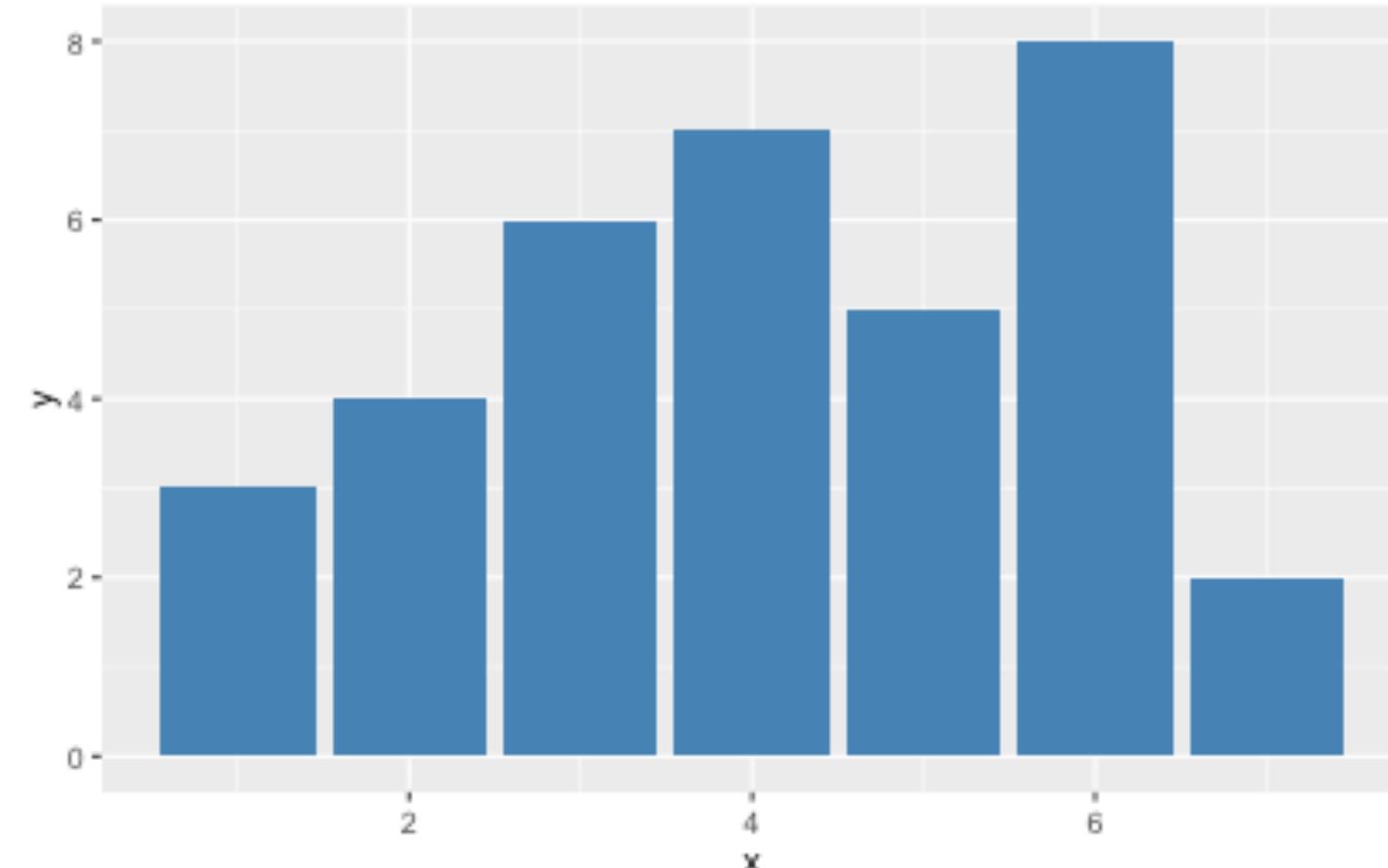
```
barchart = data.plot( kind="bar" )
```



```
barplot(data)
```

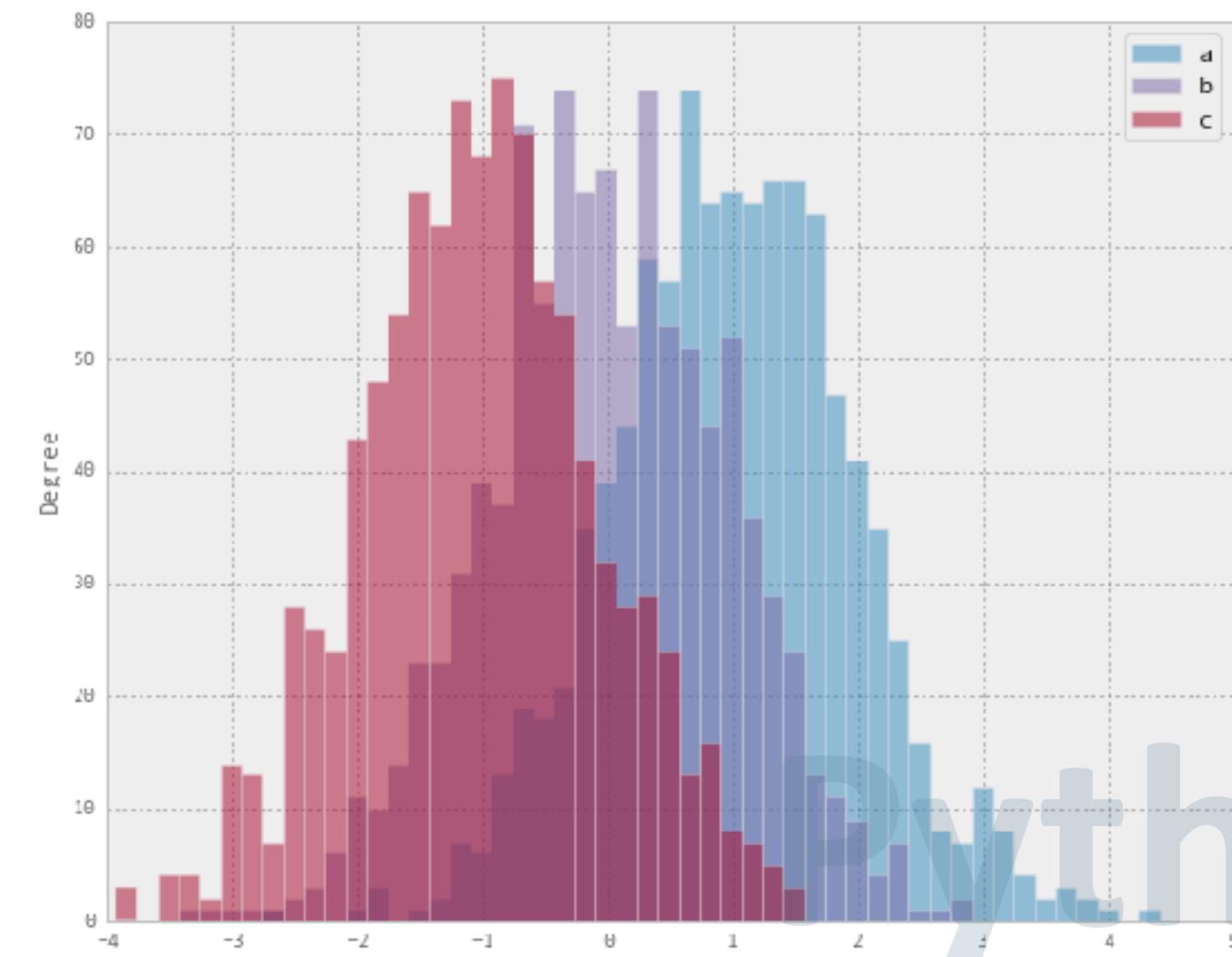


```
ggplot(dataf, aes(x, y)) +  
geom_bar(stat="identity", fill="steelblue")
```

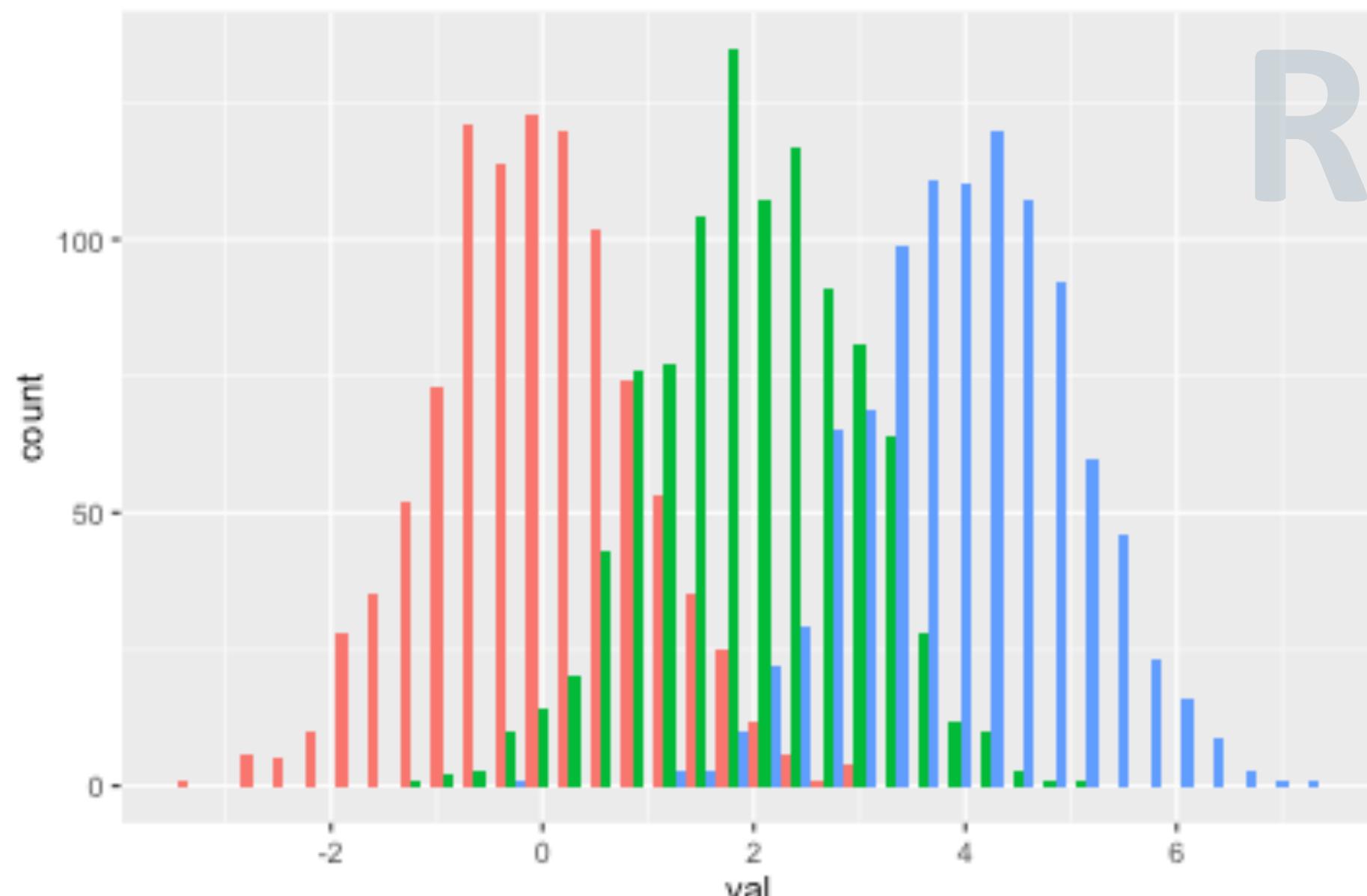


# Histograms

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



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



# Line Plots

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

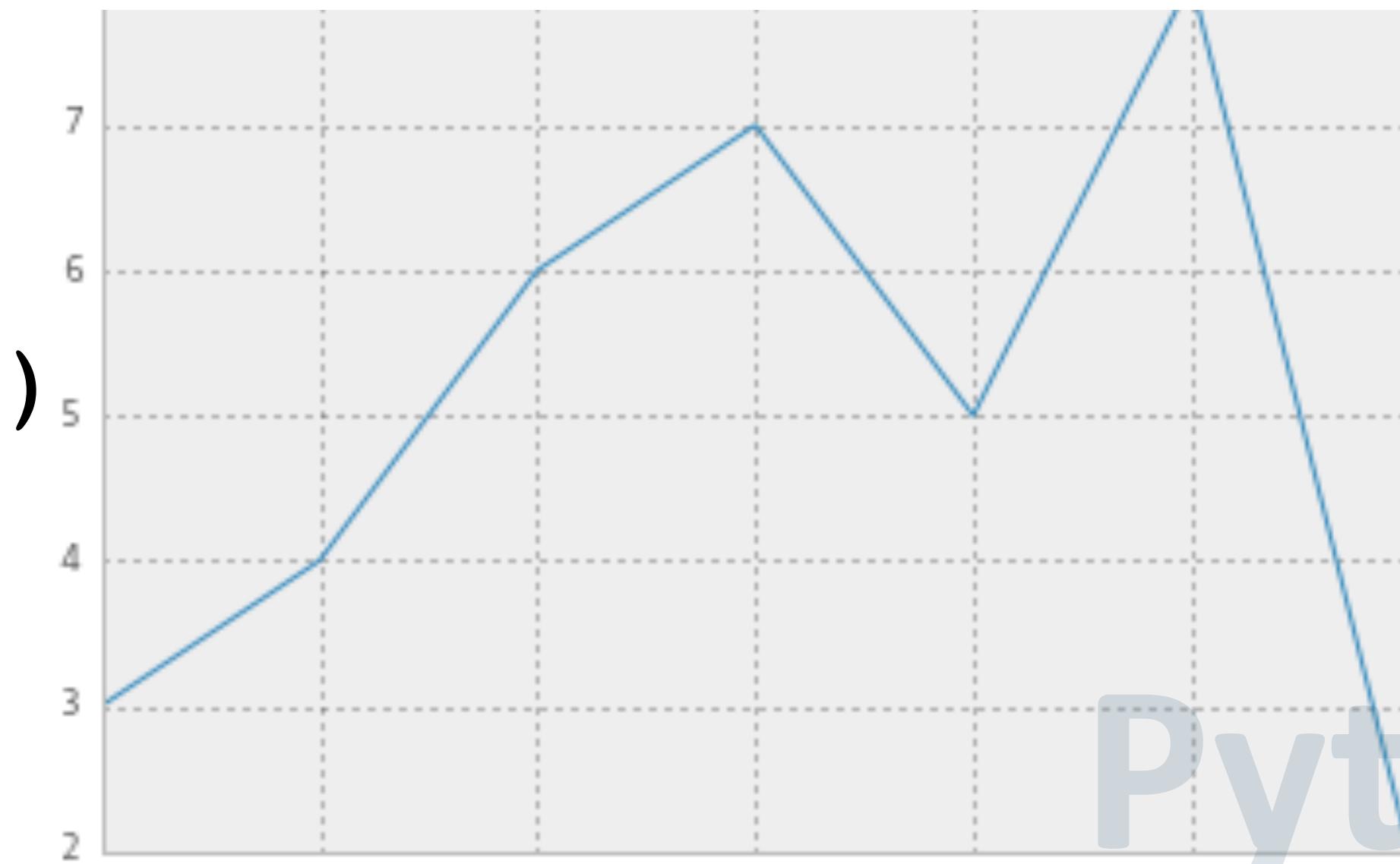
Python

R

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

# Line Plots

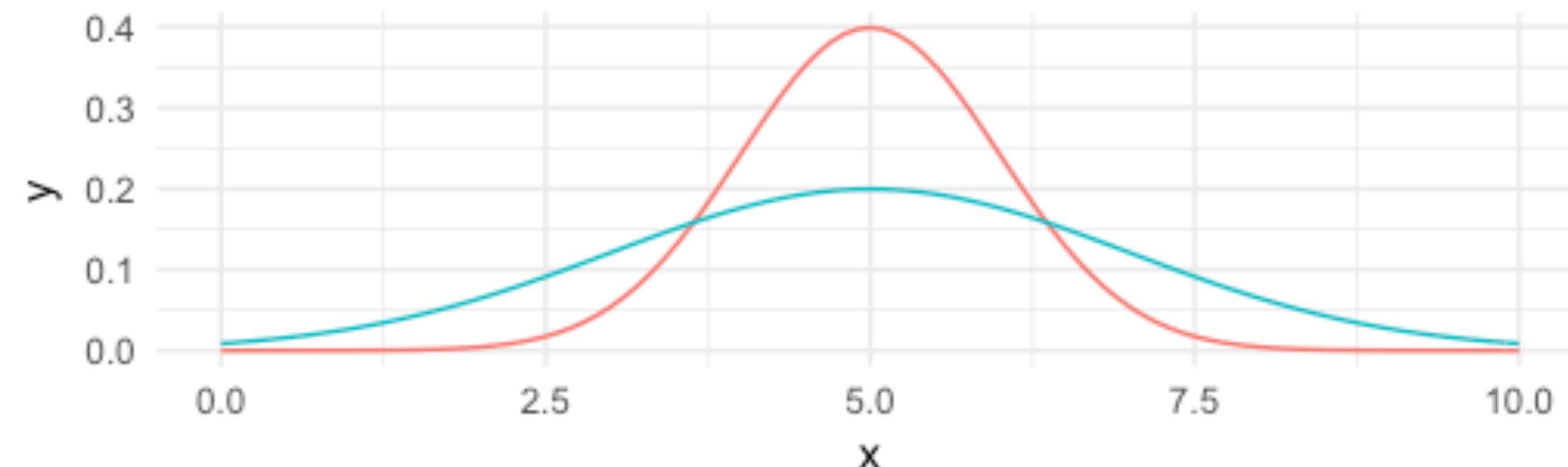
```
data = pd.Series([3,4,6,7,5,8,2])
graph = data.plot()
```



Python

```
dfy <- tibble(x=seq(0,10,0.1), y=dnorm(x, mean=5, sd=1), type="one") %>%
bind_rows(tibble(x=seq(0,10,0.1), y=dnorm(x, mean=5, sd=2), type="two"))
```

```
ggplot(dfy, aes(x, y, color=type)) +
geom_line() +
scale_color_discrete(guide=F) +
theme_minimal()
```



R

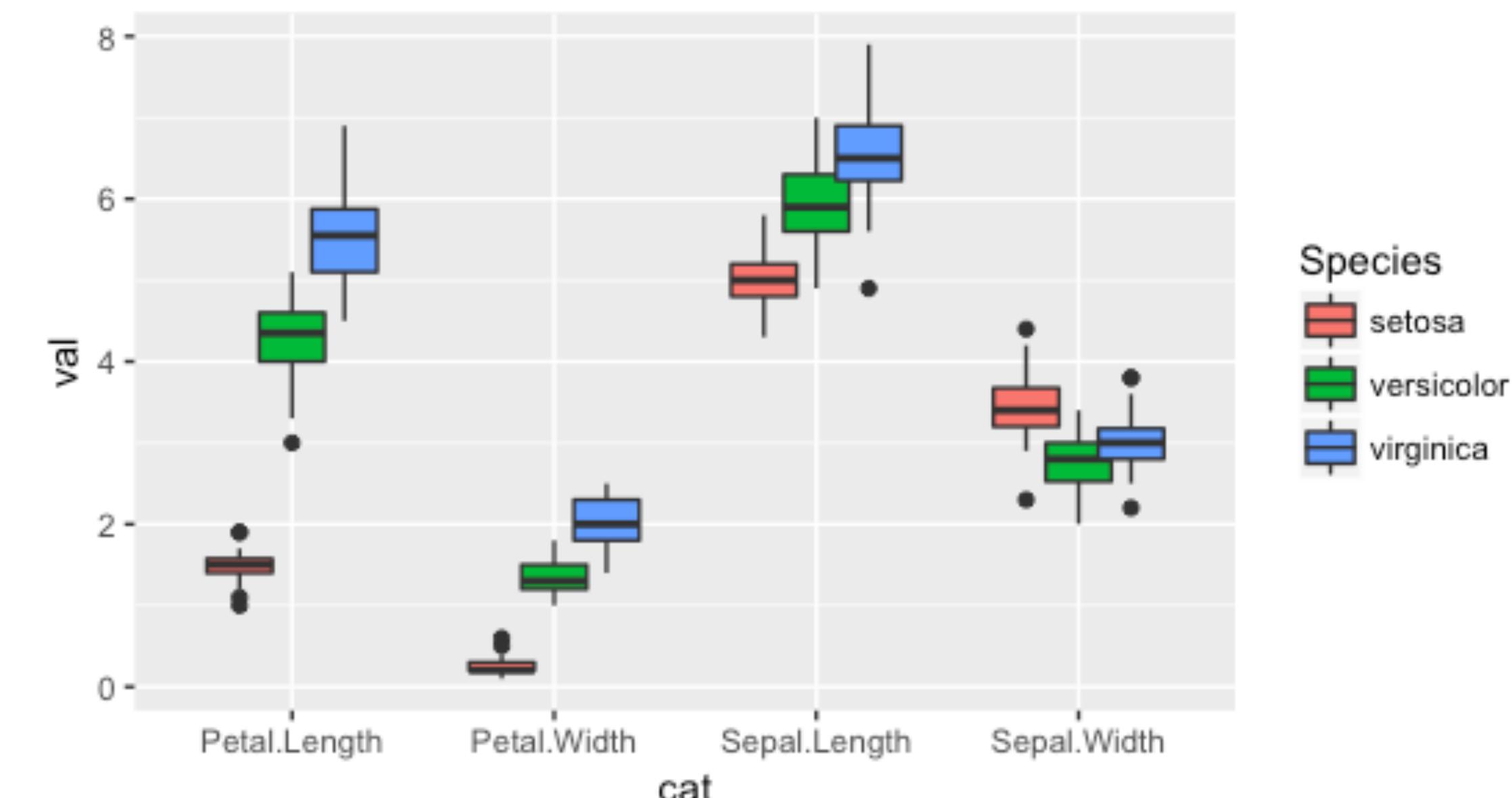
# Box Plots

```
df4.plot()
```

```
data(iris)
toplot <- iris %>% gather(cat, val, -Species)

ggplot(toplot, aes(cat, val, fill=Species)) +
+ geom_boxplot(position=position_dodge(0.6))
```

Python



O'REILLY®

# Security

BUILD BETTER DEFENSES

## Data Sources for this Training

[oreillysecuritycon.com](http://oreillysecuritycon.com)

#oreillysecurity

# 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

	<b>industry</b>	<b>hosts</b>		<b>botfam</b>	<b>orgs</b>
1	Manufacturing	256701		1 ConfickerAB	34039
2	Retail	218970		2 Sality	8591
3	Education	76455		3 Necurs	8554
4	Government/Politics	63701		4 Ramnit	7940
5	Finance	57387		5 Zeus	5394
6	Healthcare/Wellness	33915		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	host	src
1	alexa	50000	newyorkpass.com	alexa
2	bamital	10000	techcrunch.com	alexa
3	cryptolocker	10000	c92d464ec85397f9635955089ee30ff2.info	bamital
4	gameoverdga	10000	3b1976d97c9f07fc4e7b3070b3740fc4.co.cc	bamital
5	necurs	10000	ynhjqpyepyvaibh.net	cryptolocker
6	nivdort	10000	foxhndkbdseb.org	cryptolocker
			1vfglawlojizj415wzi4ulh527xy.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:** 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

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

# { LAB } time

Please complete “EDA Worksheet”



# Exercise 1: Which industry sees the most Ramnit Infections?

```
grouped_df = data[data.botfam == "Ramnit"].groupby(['industry'])
grouped_df.sum()
```

Python

R

```
dailybot %>% filter(botfam == "Ramnit") %>% count(industry, botfam, wt=hosts)
```

Exercise 2: Calculate the min, max, median and mean infected orgs across industries for each bot family, sort by median

```
group2 = data[ [ 'botfam', 'orgs' ] ].groupby( [ 'botfam' ] )
summary = group2.agg( [ np.min, np.max, np.mean, np.median, np.std ] )
summary.sort_values( [ ( 'orgs', 'median' ) ], ascending=False)
```

Python

R

```
dailybot %>%
  group_by(botfam) %>%
  summarize(min=min(orgs), max=max(orgs), median=median(orgs), mean=mean(orgs)) %>%
  arrange(desc(median))
```

Exercise 3: Which date had the total most bot infections and how many infections on that day?

```
df3 = data[ [ 'date' , 'hosts' ] ].groupby( 'date' ).sum( )
df3.sort_values(by='hosts', ascending=False).head(10)
```

Python

R

```
dailybot %>%
  count(date, wt=hosts) %>% top_n(1, n)
```

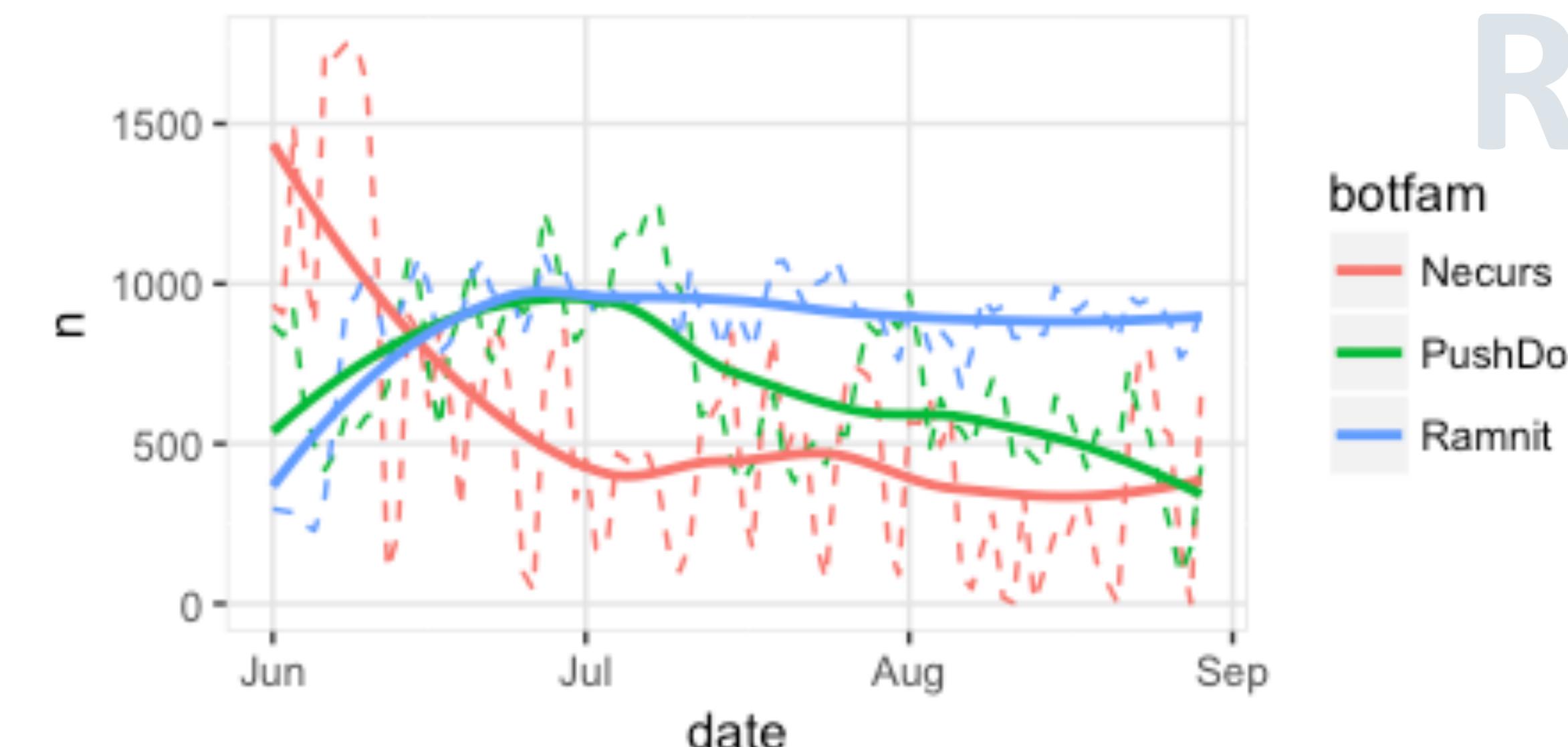
# Exercise 4: Plot the daily infected hosts for Necurs, Ramnit and Pushdo

```
filteredData = data[ data['botfam'].isin(['Necurs', 'Ramnit', 'PushDo']) ][['date', 'botfam', 'hosts']]  
groupedFilteredData = filteredData.groupby( ['date', 'botfam']).sum()  
groupedFilteredData.unstack(level=1).plot(kind='line', subplots=False)
```

Python

R

```
toplot <- dailybot %>%  
  filter(botfam %in% c("Necurs", "Ramnit", "PushDo")) %>%  
  count(date, botfam, wt=hosts)  
  
ggplot(toplot, aes(date, n, color=botfam)) +  
  geom_line(linetype="dashed") +  
  geom_smooth(se=FALSE) +  
  theme(panel.background = element_rect(fill=NA, color="gray90"),  
        panel.grid.major = element_line(color="gray90"))
```



## Exercise 5: What are the distribution of infected hosts for each day-of-week across all bot families?

Python

```
toplot <- dailybot %>%
  count(date, wt=hosts) %>%
  mutate(weekday = format(date, "%a")) %>%
  mutate(weekday = factor(weekday,
    levels=c("Sun", "Mon", "Tue", "Wed",
    "Thu", "Fri", "Sat"),
    ordered=TRUE))

ggplot(toplot, aes(weekday, n)) +
  geom_boxplot(fill="lightsteelblue") +
  scale_y_continuous("Infected Hosts", breaks = seq(5000, 12000, 1000)) +
  theme(panel.background = element_rect(fill=NA, color="gray90"),
    panel.grid.major = element_line(color="gray90"))
```

