OPEN DATA SCIENCE CONFERENCE

London | October 12th - 14th 2017

Visualizing Models

with R and Python

maxhumber



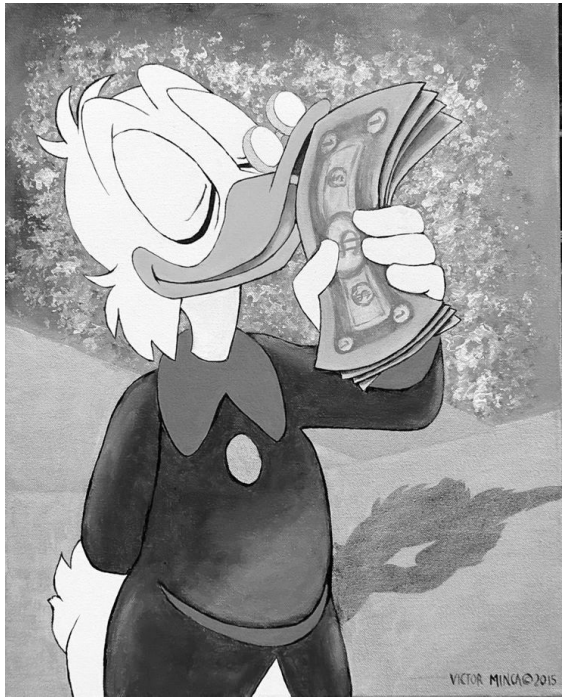
#ODSC

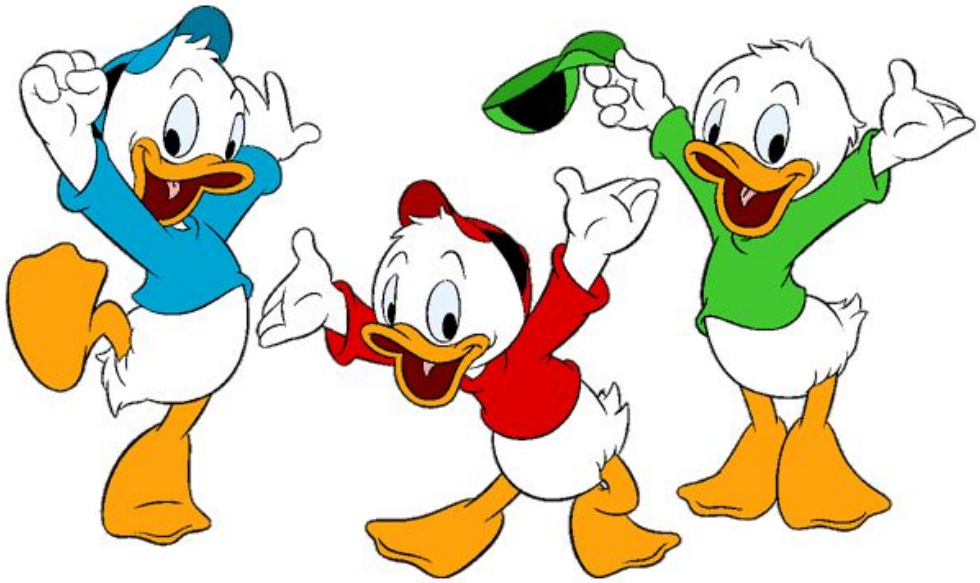
LINK TO SLIDES

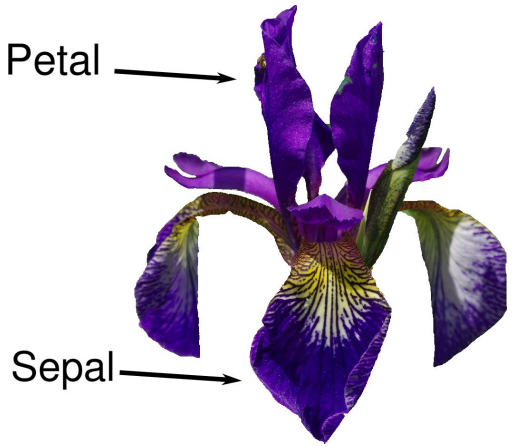
intro

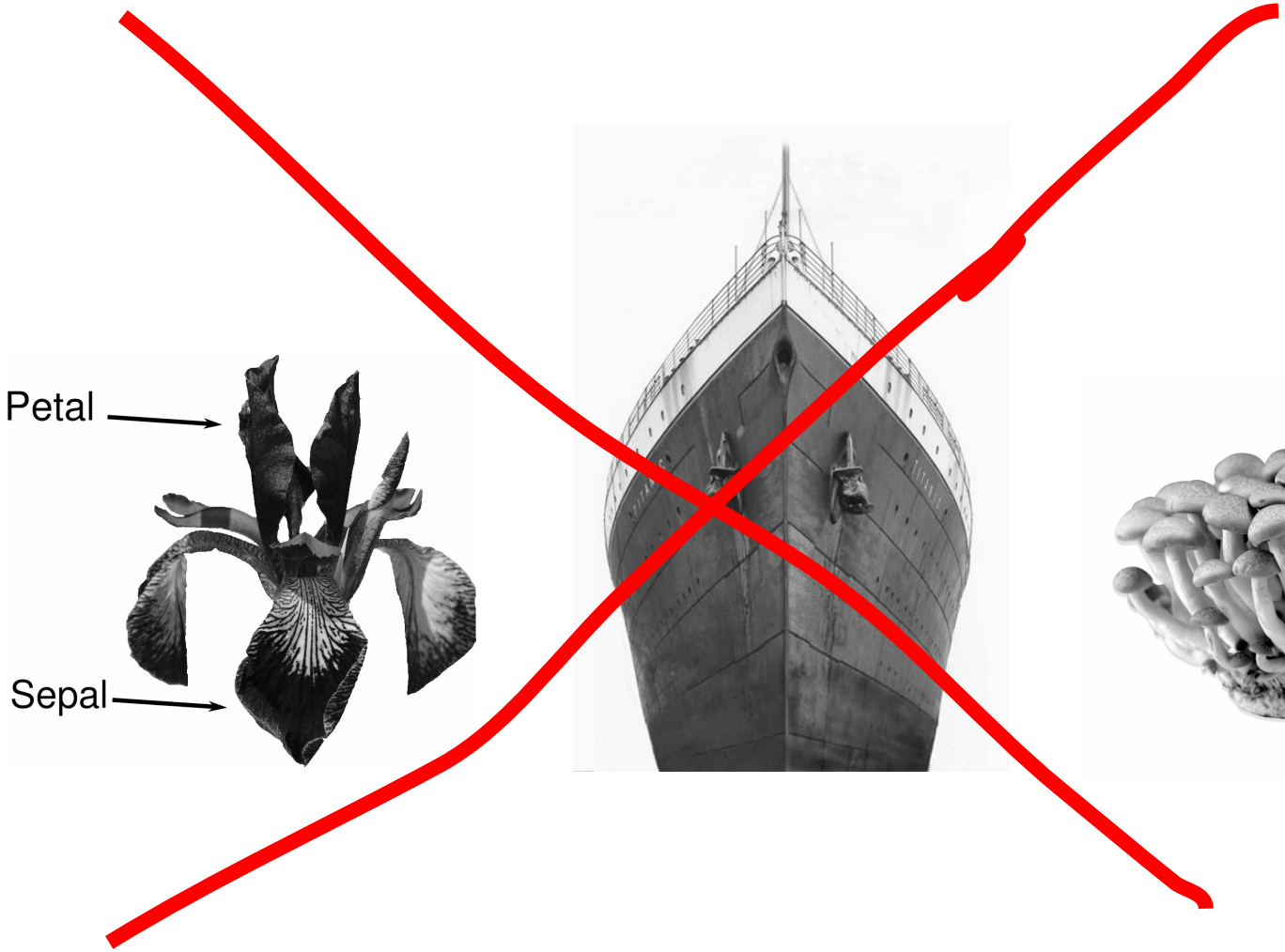

YES
NO


VICTOR MINGA©2015

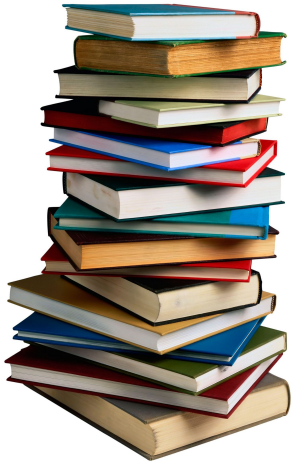

VICTOR MINGA©2015




Petal
Sepal


Petal
Sepal

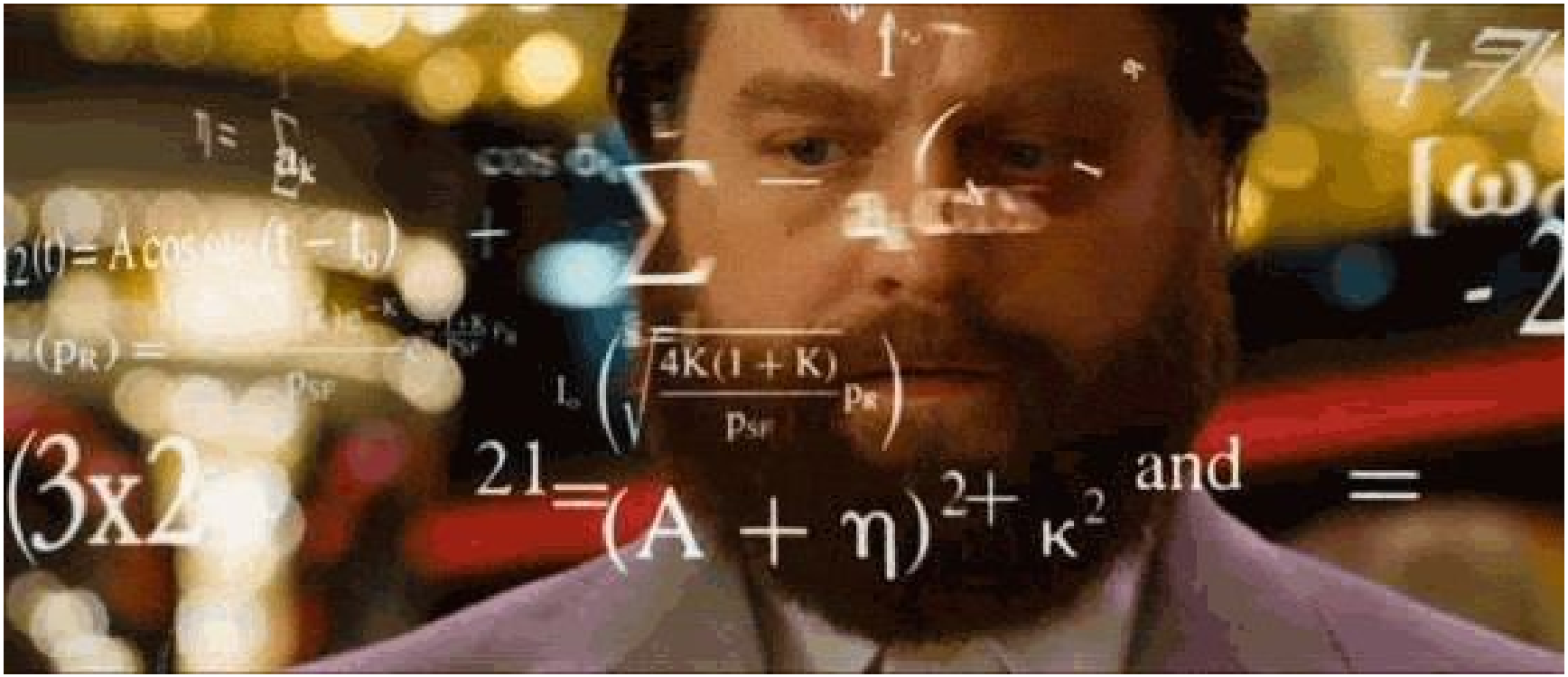

ESPN
FANTASY
FOOTBALL



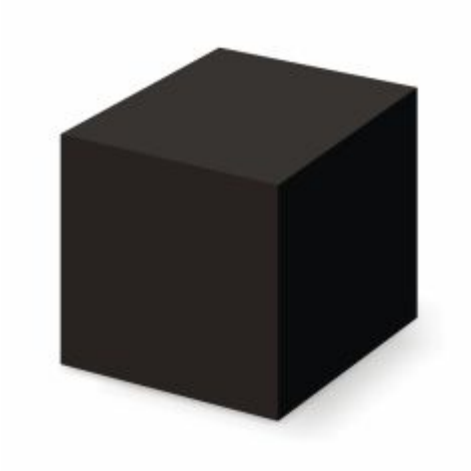
Hypothetical Outcome Plots

Separation Plots

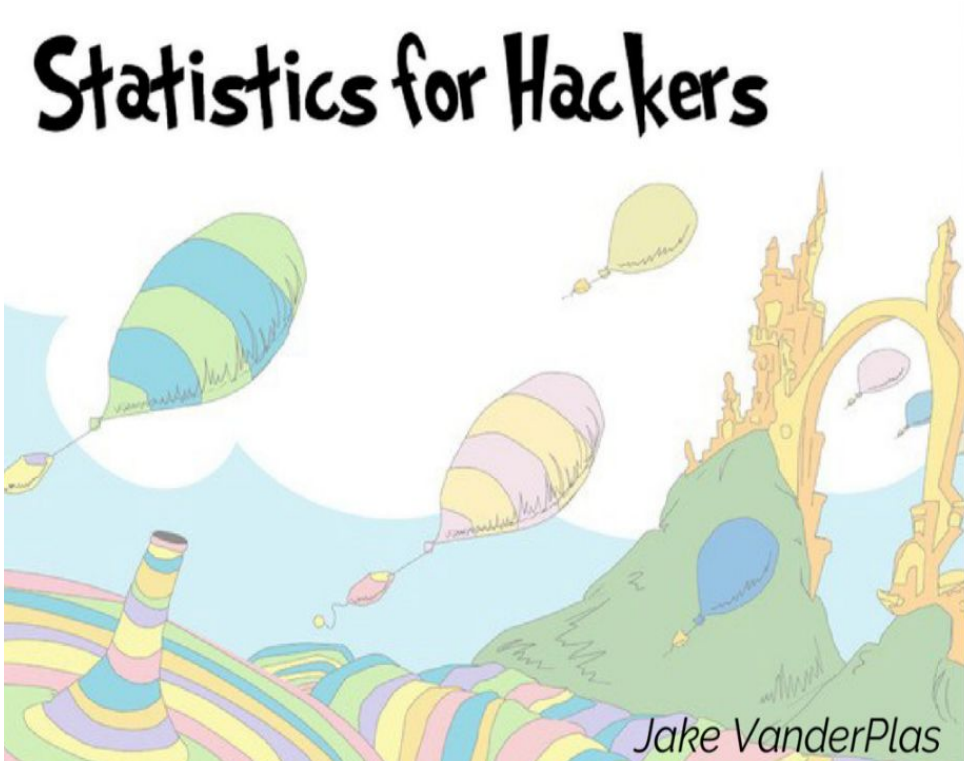
FFTrees



Animated GIF



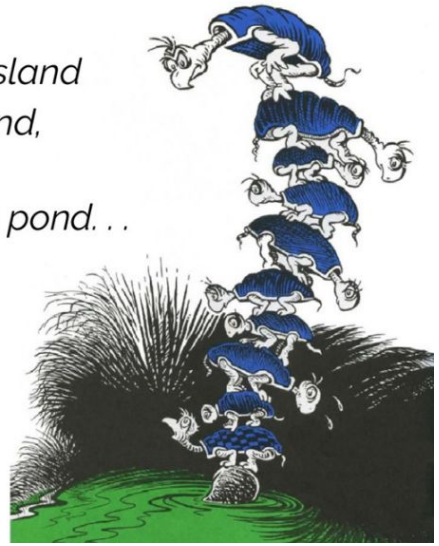
1 / 3



source: https://speakerdeck.com/jakevdp/statistics-for-hackers

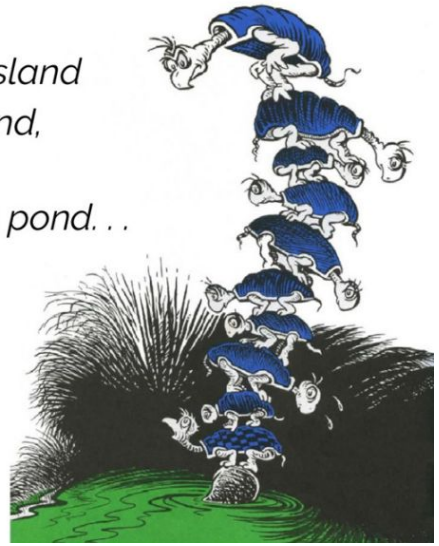
Yertle's Turtle Tower

On the far-away island
of Sala-ma-Sond,
Yertle the Turtle
was king of the pond. . .



Yertle's Turtle Tower

On the far-away island
of Sala-ma-Sond,
Yertle the Turtle
was king of the pond. . .



How High can Yertle stack his turtles?

Observe 20 of Yertle's turtle towers . . .

# of turtles										
	48	24	32	61	51	12	32	18	19	24
	21	41	29	21	25	23	42	18	23	13

- What is the mean of the number of turtles in Yertle's stack?
- What is the uncertainty on this estimate?

Classic Method:

Sample Mean:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Standard Error of the Mean:

$$\sigma_{\bar{x}} = \frac{1}{\sqrt{N}} \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2}$$

Classic Method:

Sample Mean:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Standard Error of the Mean:

$$\sigma_{\bar{x}} = \frac{1}{\sqrt{N}}\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

```
turtles <- c(
    48, 24, 51, 12,
    21, 41, 25, 23,
    32, 61, 19, 24,
    29, 21, 23, 13,
    32, 18, 42, 18
)

turtles %>% mean()
[1] 28.9

se <- function(x)
sqrt(var(x)/length(x))
turtles %>% se()
[1] 3
```

Bootstrap Resampling:

48	24	51	12
21	41	25	23
32	61	19	24
29	21	23	13
32	18	42	18

Idea:
Simulate the distribution by *drawing samples with replacement.*

Motivation:
The data estimates its own distribution – we draw random samples from this distribution.

21									

Bootstrap Resampling:

48	24	51	12
21	41	25	23
32	61	19	24
29	21	23	13
32	18	42	18

Idea:
Simulate the distribution by *drawing samples with replacement.*

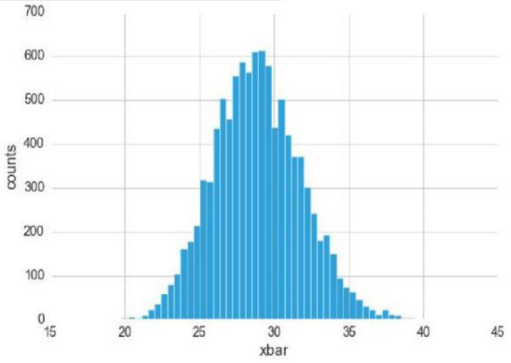
Motivation:
The data estimates its own distribution – we draw random samples from this distribution.

21	19	25	24	23	19	41	23	41	18
61	12	42	42	42	19	18	61	29	41

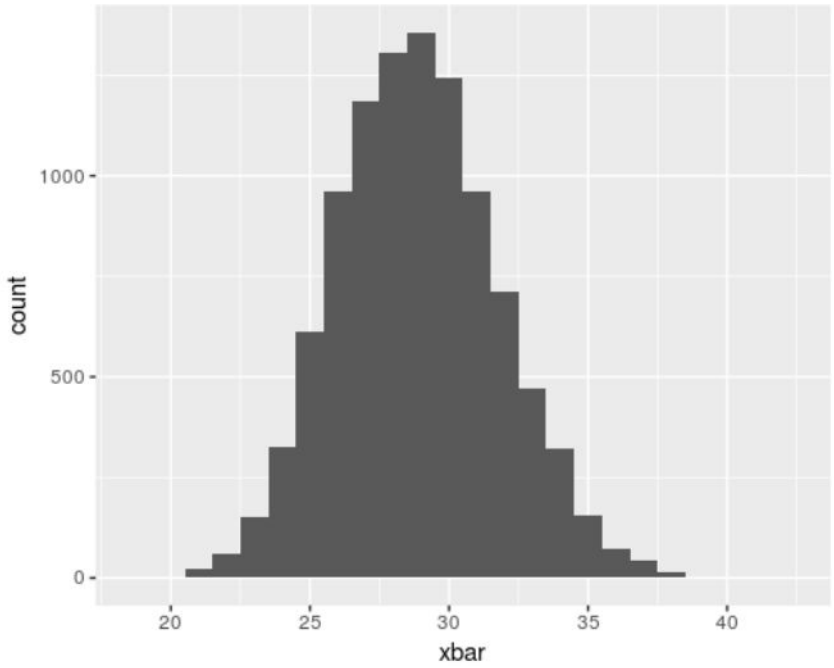
Recovers The Analytic Estimate!

```
for i in range(10000):
    sample = N[randint(20, size=20)]
    xbar[i] = mean(sample)
mean(xbar), std(xbar)
# (28.9, 2.9)
```

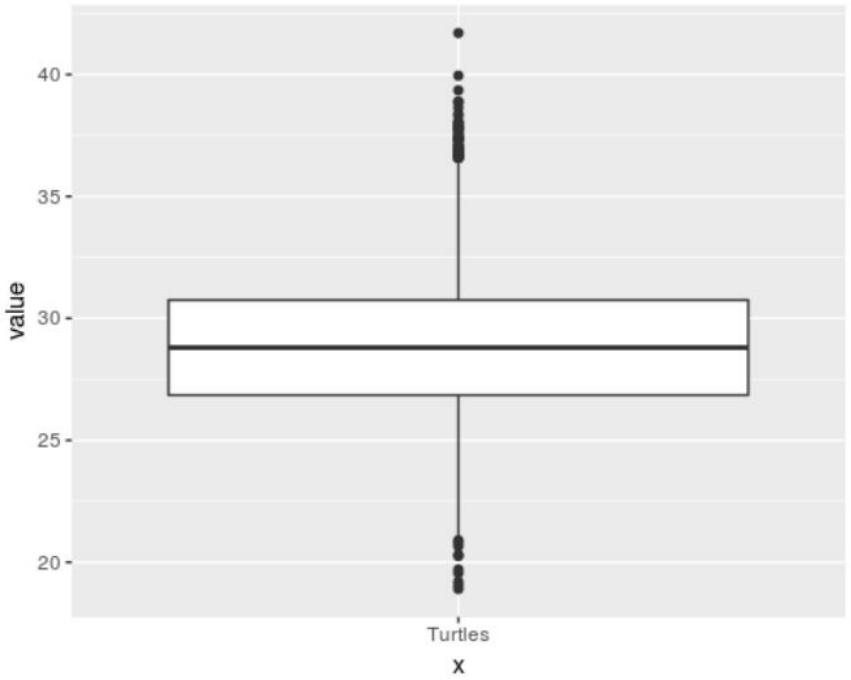
Height ≈ 29 ± 3 turtles



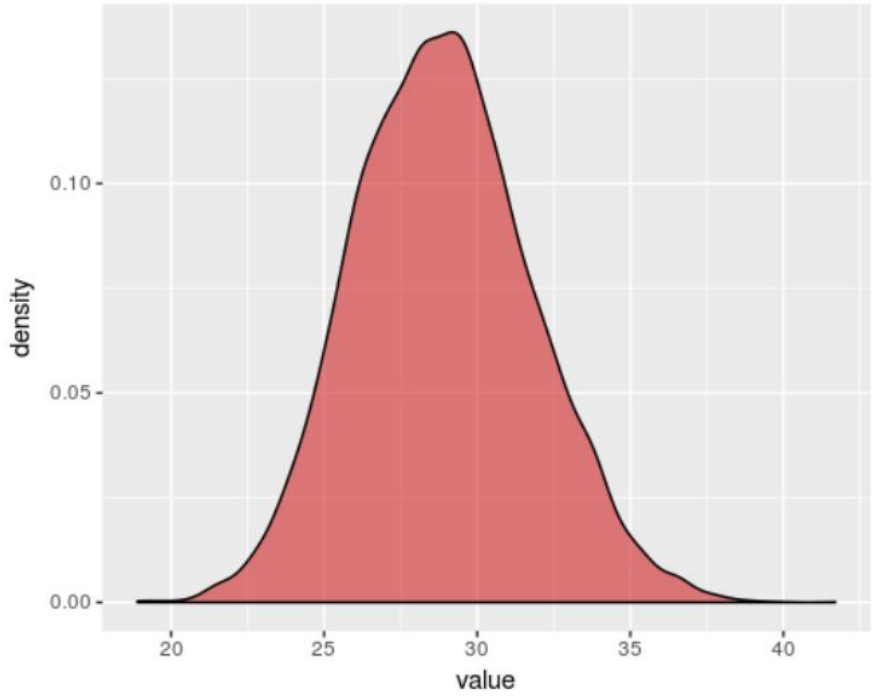

```
xbar <- numeric(10000)
for(i in 1:10000) {
    x <- sample(turtles, 20,
replace=TRUE) %>% mean()
    xbar[i] <- x
}

df <- xbar %>%
    as_data_frame() %>%
    mutate(sim = row_number())
```



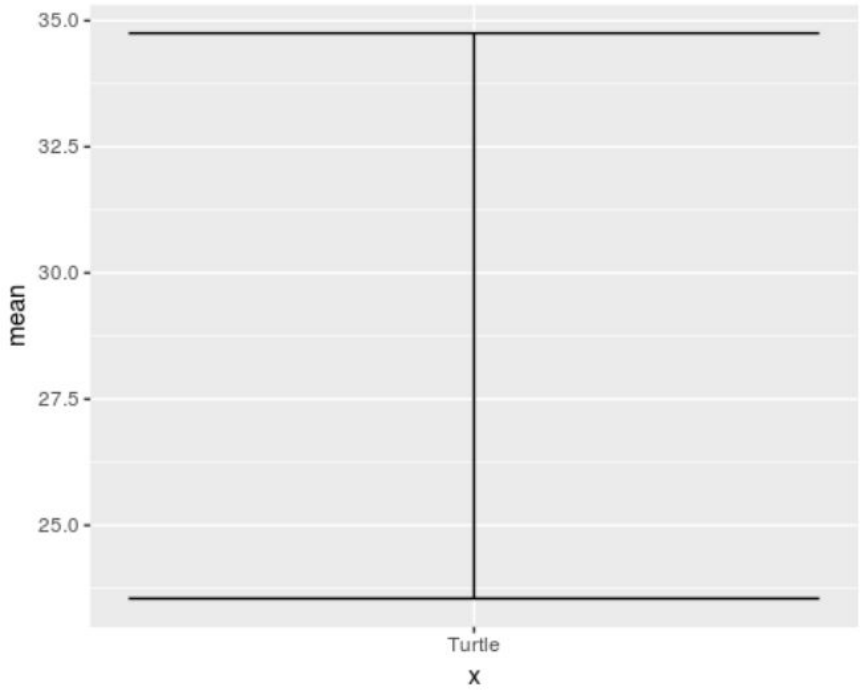

```
xbar <- numeric(10000)
for(i in 1:10000) {
    x <- sample(turtles, 20,
replace=TRUE) %>% mean()
    xbar[i] <- x
}

df <- xbar %>%
    as_data_frame() %>%
    mutate(sim = row_number())

df %>%
    ggplot(aes(x = value)) +
    geom_histogram() +
    labs(x = "xbar")
```



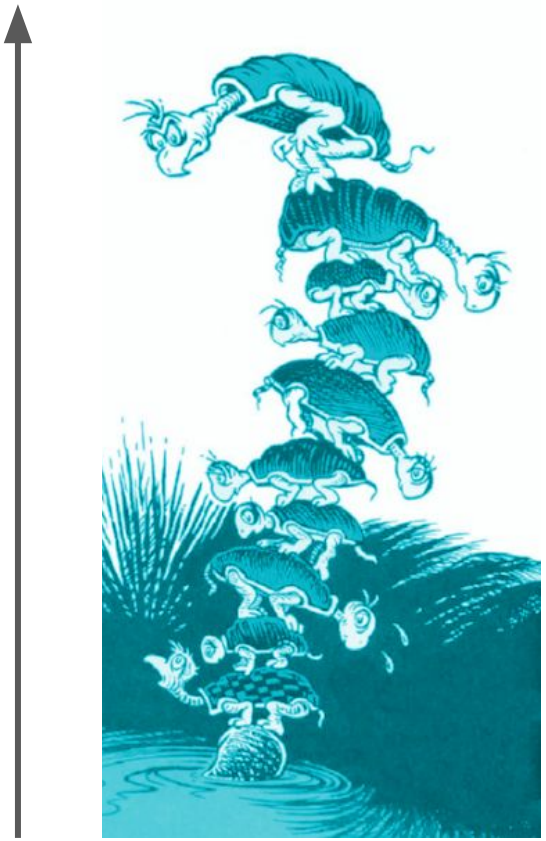

```
df %>%
    ggplot(aes(
        x = "Turtles",
        y = value)) +
    geom_boxplot()
```

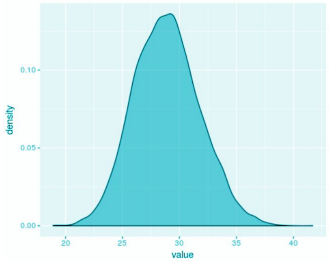


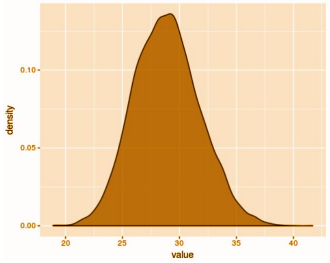

```
df %>%
    ggplot(aes(x = value)) +
    geom_density(
        fill = "#ce0000",
        alpha = 1/2)
```

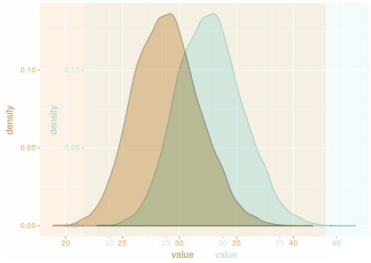



```
df %>%
    summarise(
        mean = mean(value),
        low = quantile(
           value, 0.025),
        high = quantile(
           value, 0.975)
    ) %>%
    ggplot(aes(
       x = "Turtle",
       y = mean)) +
    geom_errorbar(aes(
       ymin = low,
       ymax = high))
```




density
0.10
0.05
0.00
20
25
30
35
40
value

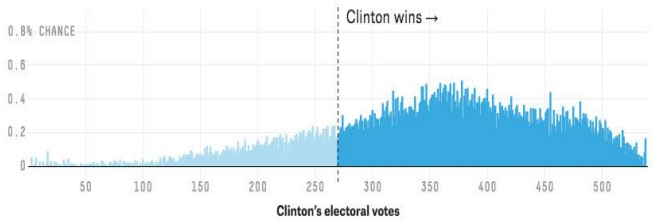

density
0.10
0.05
0.00
20
25
30
35
40
value

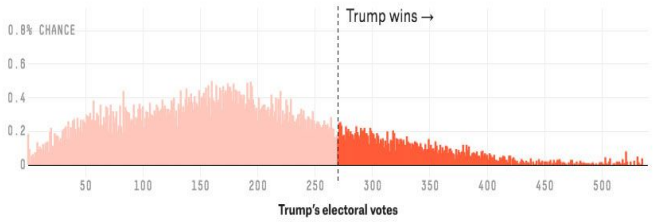

density
0.10
0.05
0.00
20
25
30
35
40
value

Chance of winning

What to expect from the Electoral College

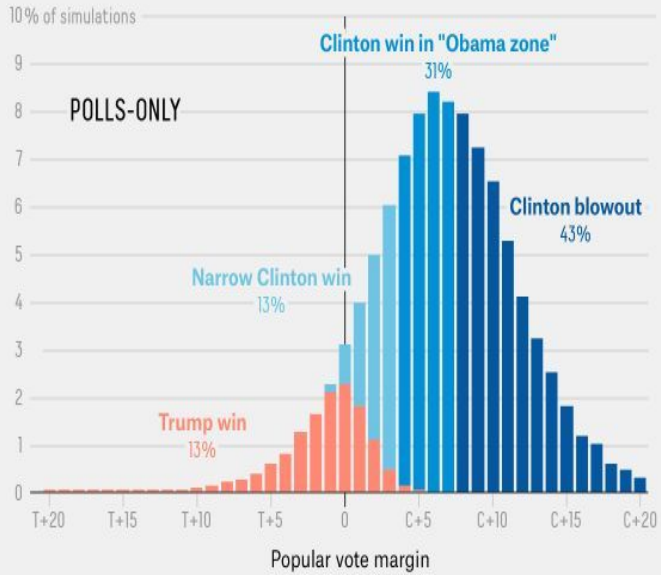
A candidate needs at least 270 electoral votes to clinch the White House. In each of our simulations, we forecast the states and note the number of electoral votes each candidate wins. That gives us a distribution for each candidate, where the tallest bar is the outcome that occurred most frequently.





Potential outcomes of the presidential election

Likelihood of scenarios given by 20,000 simulations of the FiveThirtyEight polls-only and polls-plus election forecasts




ESPN
FANTASY
FOOTBALL

Animated GIF

https://speakerdeck.com/maxhumber/webscraping-with-rvest-and-purrr

Max	LOLBAYES		SureWin	Elvina
2-1-0 (6), Streak: **W1**	**0.00**	VS	**0.00**	**2-1-0 (5)**, Streak: **W2**
Waiver: 5				Waiver: 6

Hide Chart

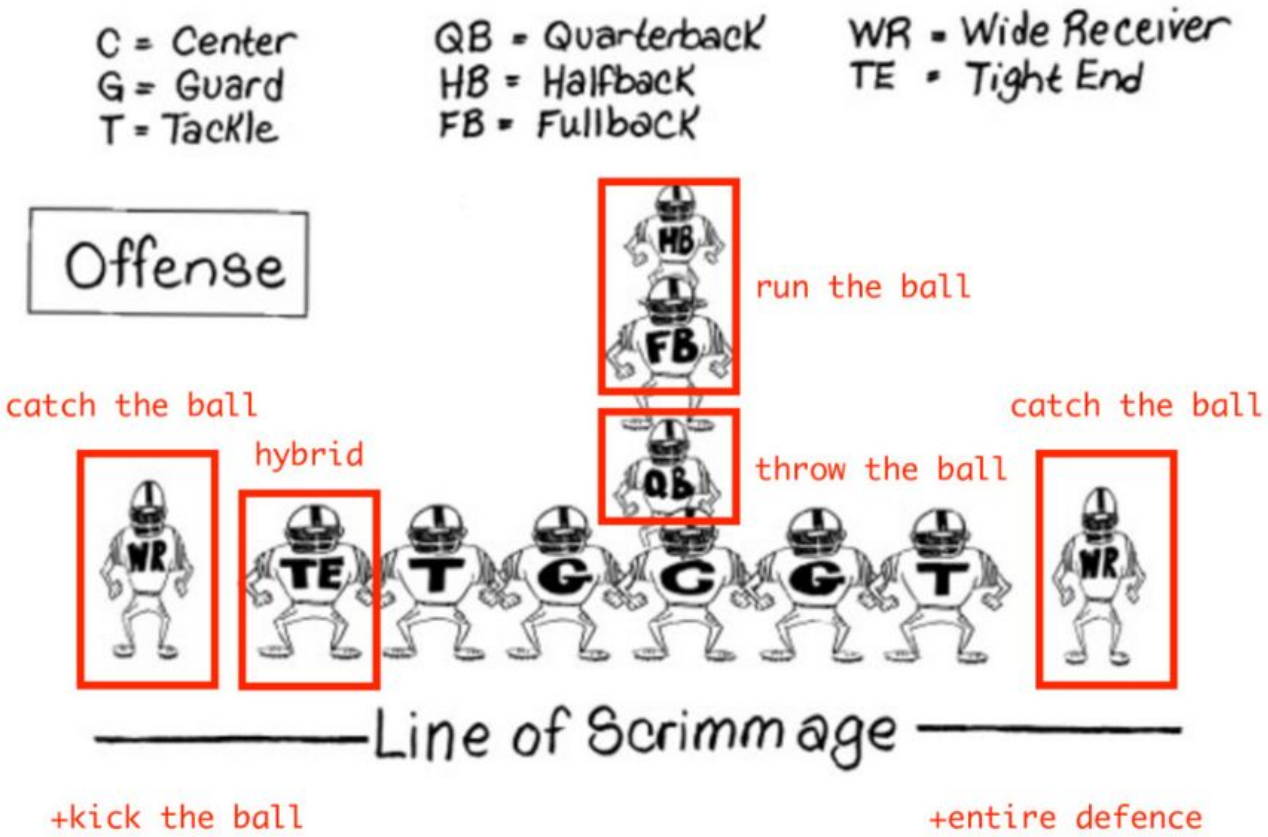
91.43

QB-1	QB-2	RB-1	RB-2	WR-1	WR-2	WR-3	TE	W/R	K	DEF
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Hide Chart

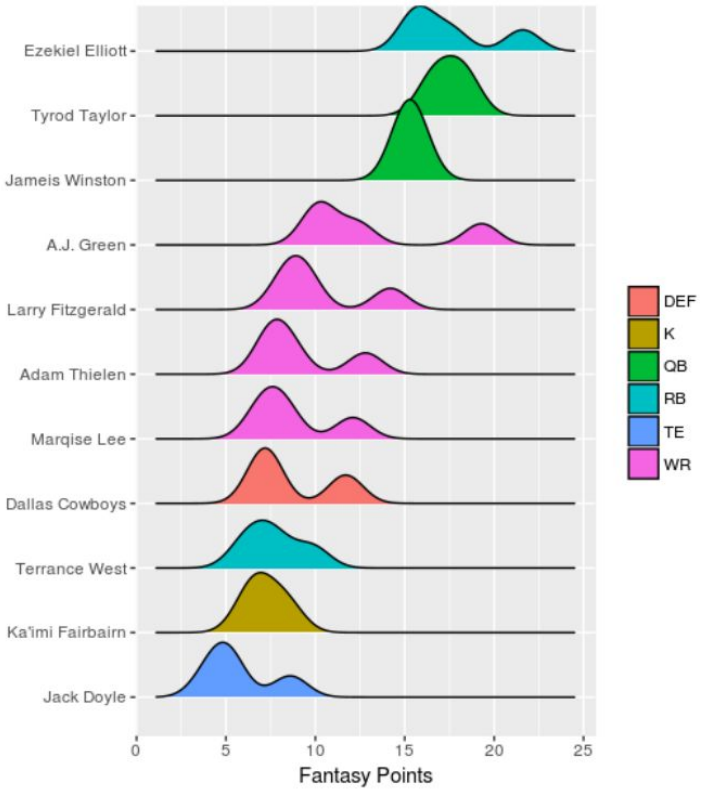
85.88

QB-1	QB-2	RB-1	RB-2	WR-1	WR-2	WR-3	TE	W/R	K	DEF
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00



position	name	week	points	source
qb	All			All
QB	Aaron Rodgers	4	22.22	NFL.com
QB	Aaron Rodgers	4	21.30	Fantasy Pros
QB	Aaron Rodgers	4	21.80	ESPN
QB	Alex Smith	4	18.39	NFL.com
QB	Alex Smith	4	16.20	Fantasy Sharks
QB	Alex Smith	4	16.80	Fantasy Pros
QB	Alex Smith	4	15.90	ESPN
QB	Andy Dalton	4	14.81	NFL.com
QB	Andy Dalton	4	14.60	Fantasy Sharks
QB	Andy Dalton	4	16.30	Fantasy Pros
QB	Andy Dalton	4	15.80	ESPN
QB	Ben Roethlisberger	4	16.63	NFL.com
QB	Ben Roethlisberger	4	14.80	Fantasy Sharks
QB	Ben Roethlisberger	4	15.00	Fantasy Pros
QB	Ben Roethlisberger	4	15.40	ESPN

#1
#2
#3
#4




```
df %>%
    filter(name %in% home) %>%
    ggplot(aes(
        x = points,
        y = reorder(name, points),
        fill = position)) +
    geom_density_ridges(
        scale = 1.25, alpha = 1) +
    labs(y = "", x = "Fantasy Points")
```

```
df <- read_csv("df.csv")

home <- c(
    "Tyrod Taylor", "Jameis Winston",
    "Terrance West", "Ezekiel Elliott",
    "A.J. Green", "Larry Fitzgerald", "Adam Thielen",
    "Marqise Lee",
    "Jack Doyle",
    "Ka'imi Fairbairn",
    "Dallas Cowboys"
)

away <- c(
    "Matthew Stafford", "Jared Goff",
    "DeMarco Murray", "Jordan Howard",
    "Demaryius Thomas", "Sammy Watkins", "Jamison Crowder",
    "Eric Ebron",
    "Chris Carson",
    "Steven Hauschka",
    "New England Patriots"
)
```

```r
sim <- function(df=df, players) {

    points <- df %>%
        filter(name %in% players) %>%
        group_by(name) %>%
        sample_n(1, replace = TRUE) %>%
        ungroup() %>%
        summarise(total = sum(points)) %>%
        pull(total)

    return(points)
}
```

```
sim <- function(df=df, players) {

    points <- df %>%
        filter(name %in% players) %>%
        group_by(name) %>%
        sample_n(1, replace = TRUE) %>%
        ungroup() %>%
        summarise(total = sum(points)) %>%
        pull(total)

    return(points)
}

sim(df, home)
[1] 126.14
```

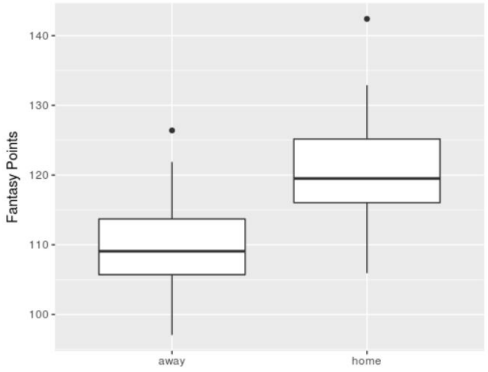
```
sim <- function(df=df, players) {

    points <- df %>%
        filter(name %in% players) %>%
        group_by(name) %>%
        sample_n(1, replace = TRUE) %>%
        ungroup() %>%
        summarise(total = sum(points)) %>%
        pull(total)

    return(points)
}

sim(df, home)
[1] 126.14
sim(df, away)
[1] 103.52
```

	value	team	sim
1	115.45	home	1
2	107.92	away	1
3	119.11	home	2
4	107.23	away	2
5	116.17	home	3
6	116.14	away	3
7	123.24	home	4
8	107.93	away	4
9	125.07	home	5
10	112.36	away	5
11	128.42	home	6
12	109.11	away	6
13	116.32	home	7
14	107.63	away	7

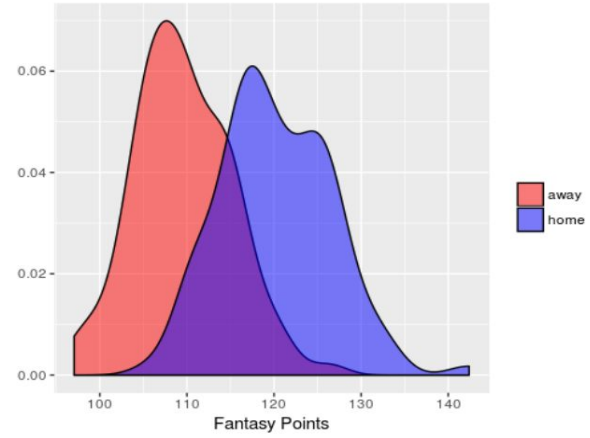
```
sim_home <- replicate(100, sim(df, home))
sim_away <- replicate(100, sim(df, away))
```

	value	team	sim
1	115.45	home	1
2	107.92	away	1
3	119.11	home	2
4	107.23	away	2
5	116.17	home	3
6	116.14	away	3
7	123.24	home	4
8	107.93	away	4
9	125.07	home	5
10	112.36	away	5
11	128.42	home	6
12	109.11	away	6
13	116.32	home	7
14	107.63	away	7

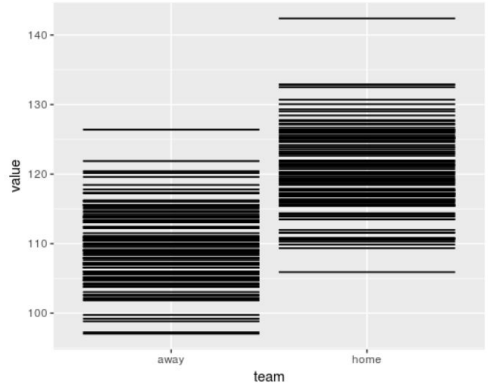
```
sim_home <- replicate(100, sim(df, home))
sim_away <- replicate(100, sim(df, away))

sim_home <- sim_home %>%
    as_data_frame() %>%
    mutate(team = "home")

sim_away <- sim_away %>%
    as_data_frame() %>%
    mutate(team = "away")

sim_all <- bind_rows(sim_home, sim_away) %>%
    group_by(team) %>%
    mutate(sim = row_number())
```




```
sim_all %>%
    ggplot(aes(y = value, x = team)) +
    geom_boxplot() +
    labs(x = "", y = "Fantasy Points")
```




```
sim_all %>%
    ggplot(aes(x = value, fill = team)) +
    geom_density(alpha = 1/2) +
    scale_fill_manual(
        values = c("red", "blue")) +
    labs(y = "", x = "Fantasy Points")
```

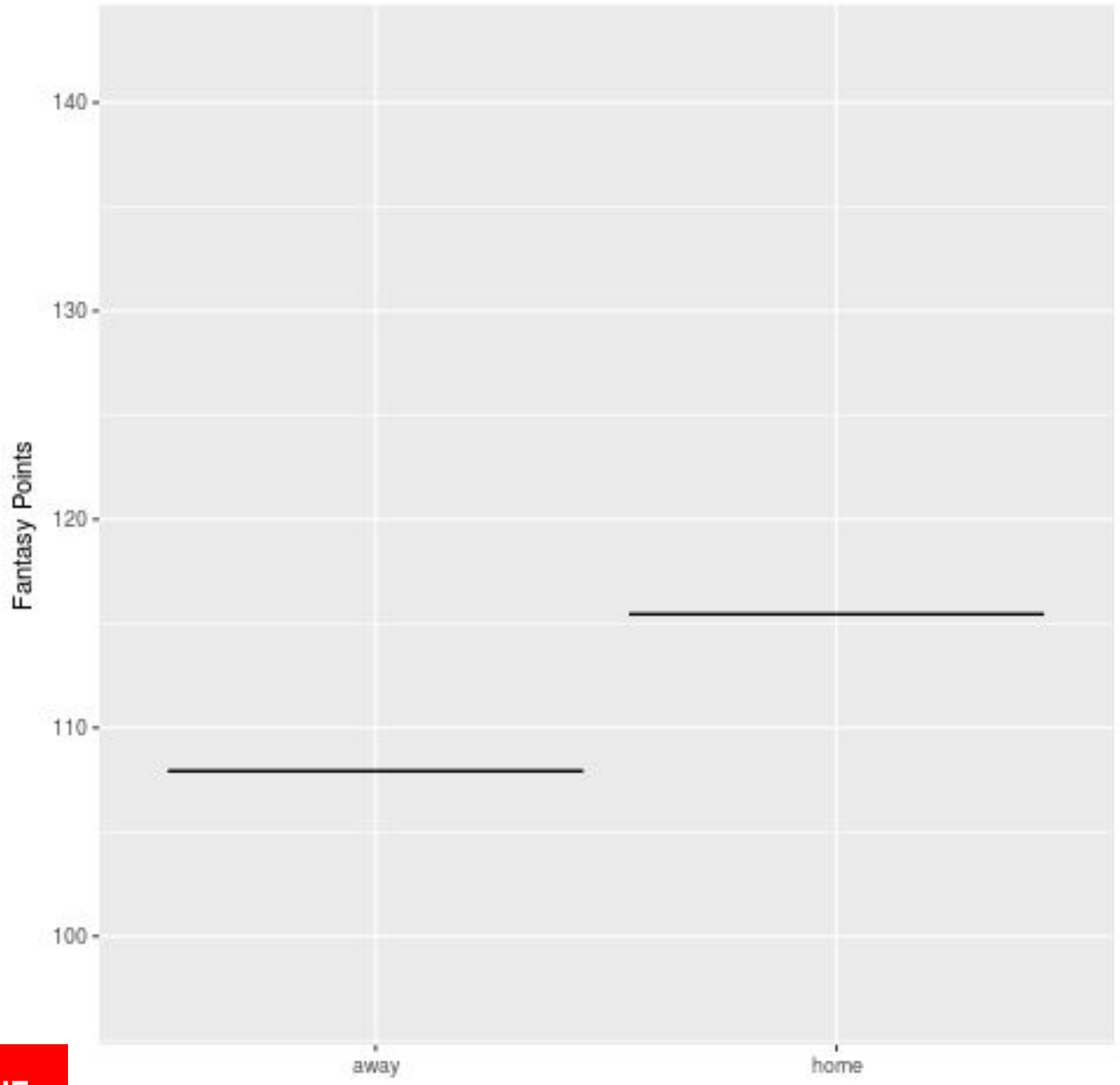



```
sim_all %>%
    ggplot(aes(x = team, y = value)) +
    geom_errorbar(aes(
        ymin = value, ymax = value)) +
    labs(x = "", y = "Fantasy Points")
```

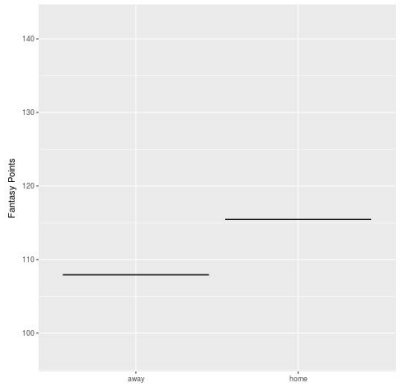
Jessica Hullman, Paul Resnick and Eytan Adar

Rather than showing a continuous probability distribution, HOPs visualize a set of draws from a distribution, where each draw is shown as a new plot in either a small multiples or animated form. HOPs enable a user to experience uncertainty in terms of countable events, just like we experience probability in our day to day lives.

Source: https://medium.com/hci-design-at-uw/hypothetical-outcomes-plots-experiencing-the-uncertain-b9ea60d7c740

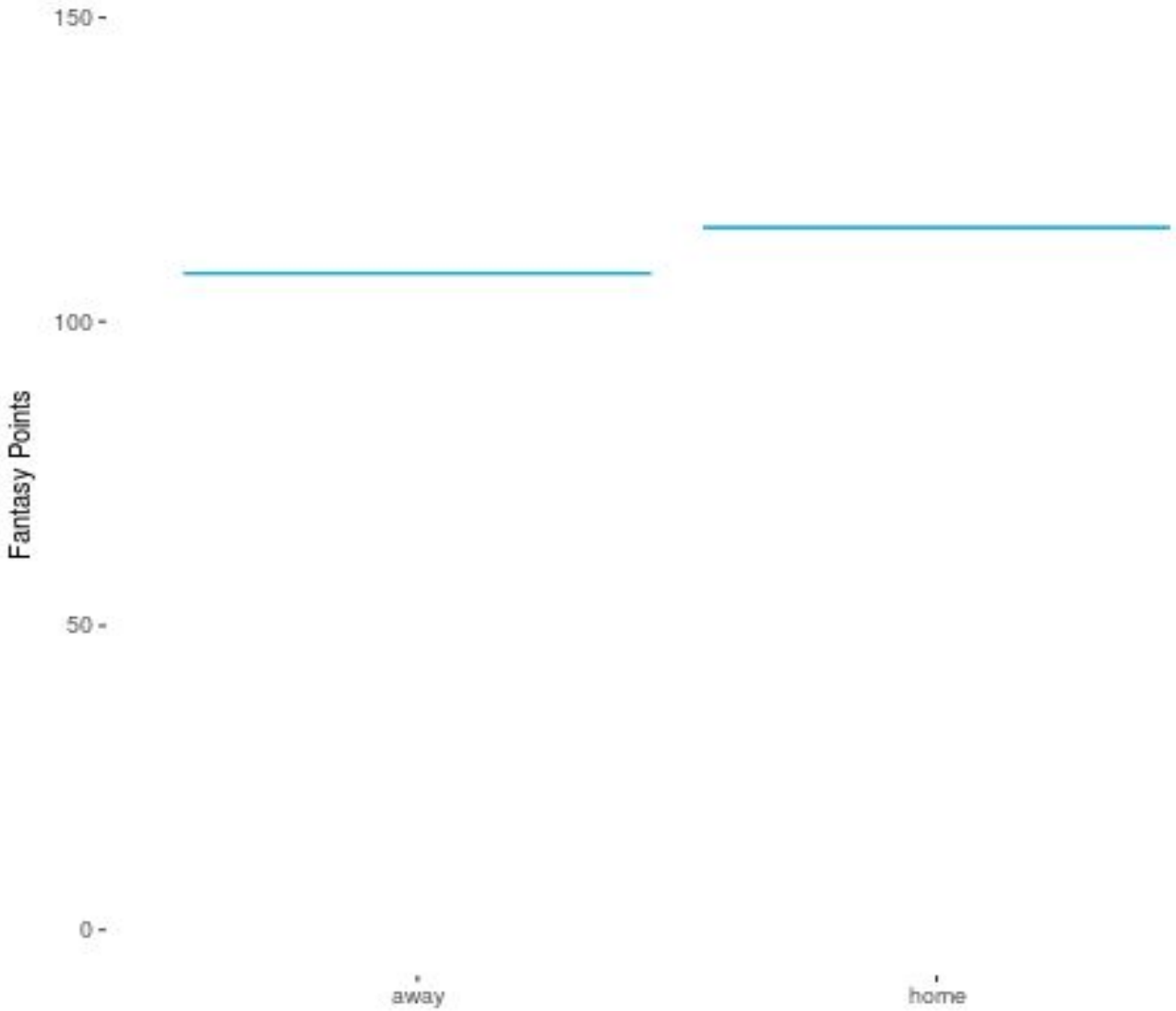


Animated GIF



```
p <- sim_all %>%
    ggplot(aes(x = team, y = value, frame = sim)) +
    geom_errorbar(aes(ymin = value, ymax = value)) +
    labs(x = "", y = "Fantasy Points")

gganimate(p, title_frame = FALSE)
```

```
p <- sim_all %>%
    ggplot(aes(x = team, y = value)) +
    geom_errorbar(aes(ymin = value, ymax = value,
        frame = sim, cumulative = TRUE),
        color = "grey80", alpha = 1/8) +
    geom_errorbar(aes(
        ymin = value, ymax = value, frame = sim),
        color = "#00a9e0") +
  scale_y_continuous(limits = c(0, 150)) +
  theme(panel.background = element_rect(fill = "#FFFFFF")) +
  labs(title = "", y = "Fantasy Points", x = "")

gganimate(p, title_frame = FALSE)
```



Animated GIF

GAME CENTER

Fantasy NFL — Week 5 — 1 2 3 4 **5** 6 7 8 9 10 11 12 13

LOLBAYES	99.10	**TacoCorp**	**129.82**	Argos	103.44	2LEGIT2QUIT	66.38	YoMamma	67.72	**Inadequately Inf…**	**123.90**	**Humabluea-Pua-a**	**114.00**
SureWin	**102.72**	Cab Sauvage	52.98	**Palumbro**	**117.34**	**Forgetting Bran…**	**99.88**	**BillNyeMoneyba…**	**106.06**	Chipotle	117.06	Team 13	91.40

Max
3-2-0 (6), Streak: **L1**
Waiver: 8

LOLBAYES 99.10 — FINAL — SureWin 102.72

Elvina
4-1-0 (2), Streak: **W4**
Waiver: 10

Hide Chart

99.10

QB-1	QB-2	RB-1	RB-2	WR-1	WR-2	WR-3	TE	W/R	K	DEF
8.94	18.46	2.90	13.20	22.90	5.10	4.50	10.90	2.20	2.00	8.00

Hide Chart

102.72

QB-1	QB-2	RB-1	RB-2	WR-1	WR-2	WR-3	TE	W/R	K	DEF
15.64	2.58	3.00	9.80	5.40	16.50	18.20	2.40	14.20	9.00	6.00

2/3

Animated GIF

```
def create_data():
    N = 1000
    x1 = np.random.normal(loc=0, scale=1, size=N)
    x2 = np.random.normal(loc=0, scale=1, size=N)
    x3 = np.random.randint(2, size=N) + 1
    # linear combination
    z = 1 + 2*x1 + -3*x2 + 0.5*x3
    # inv-logit function
    pr = [1 / (1 + np.exp(-i)) for i in z]
    y = np.random.binomial(1, p=pr, size=N)
    return y, x1, x2, x3
```

```
df.head(5)
```

	x1	x2	x3	y
0	-0.177208	0.034655	1	0
1	-0.252356	-1.340353	2	1
2	-0.202379	-0.400502	2	1
3	0.517305	-0.504932	2	1
4	-0.513989	0.344414	1	0

```
np.random.seed(1993)
y, x1, x2, x3 = create_data()

df = pd.DataFrame({
    'y':y,
    'x1':x1,
    'x2':x2,
    'x3':x3
})

df.head(5)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

X = df[['x1', 'x2', 'x3']]
y = df['y']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.8, random_state=0)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

X = df[['x1', 'x2', 'x3']]
y = df['y']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.8, random_state=0)

model = LogisticRegression()
model.fit(X_train, y_train)
```

```
from sklearn.metrics import accuracy_score, roc_auc_score

predicted = model.predict(X_test)
probs = model.predict_proba(X_test)

print("Accuracy:", accuracy_score(y_test, predicted))
print("AUC:", roc_auc_score(y_test, probs[:, 1]))
```

```
Accuracy: 0.89
AUC: 0.92
```

```
from sklearn.met                                    roc_auc_score

predicted = mode
probs = model.pr

print("Accuracy:                                   redicted))
print("AUC:", ro                                     1]))
```

```
Accuracy: 0.89
AUC: 0.92
```

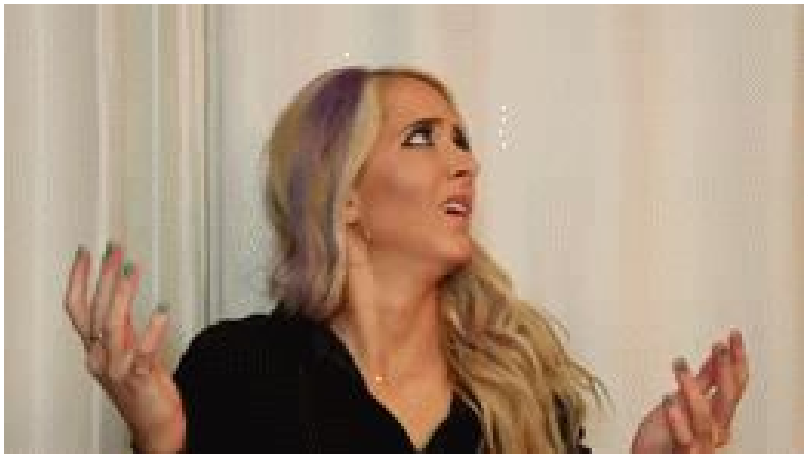
Animated GIF

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

expected = y_test
predicted = model.predict(X_test)

print(classification_report(expected, predicted))
```

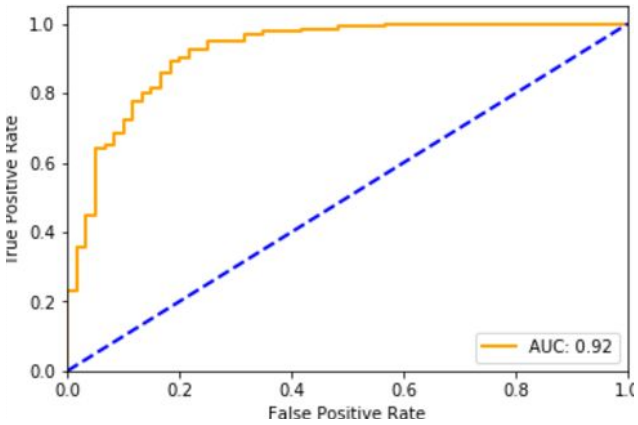
```
precision    recall  f1-score   support

          0       0.87      0.75      0.80         60
          1       0.90      0.95      0.92        140

avg / total       0.89      0.89      0.89        200
```

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

expected = y_test
predicted = model.predict(X_test)

print(classification_report(expected, predicted))
```

```
precision

          0                   80          60
          1                   92         140

avg / total                   89         200
```

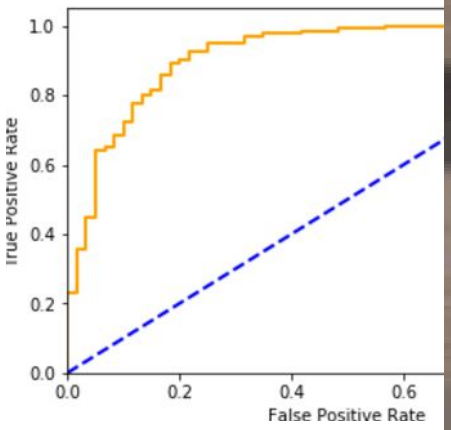


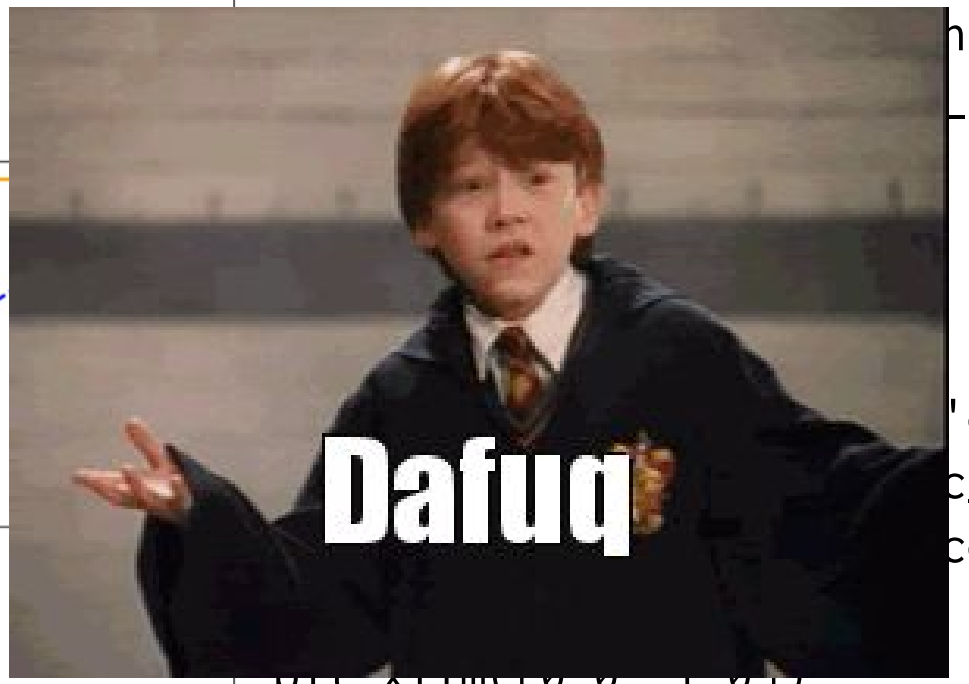
Animated GIF




```
# roc curves
from sklearn.metrics import roc_curve, auc
y_score = model.fit(X_train,
y_train).decision_function(X_test)
fpr, tpr, _ = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='orange', lw=lw,
label='AUC: {}'.format(roc_auc))
plt.plot([0, 1], [0, 1], color='blue',
lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show();
```

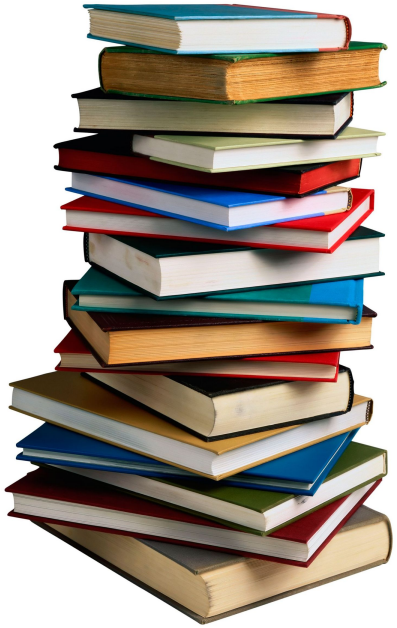
```
# roc curves
from sklearn.metrics import roc_curve, auc
y_score = model.fit(X_train,
                    n(X_test)
                    _test, y_score)
```






```
                    'orange', lw=lw,
                    c_auc))
                    color='blue',
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show();
```

Animated GIF



2017 READING CHALLENGE

Participants	**2,695,998**
Books Pledged	**122,201,601**
Books Finished	**33,839,764**
Avg. Books Pledged	**45**
Challenges Completed	**6,819**
Time Left	**92 days, 7 hours**

You have read 38 of 52 books. Edit

73%

You're on track!

View all »

Like · Comment · View Books · Share:

FRIENDS **COMMUNITY**

2016 Reading Challenge

Congrats! You read 52 books of your goal of 52!

100%

2015 Reading Challenge

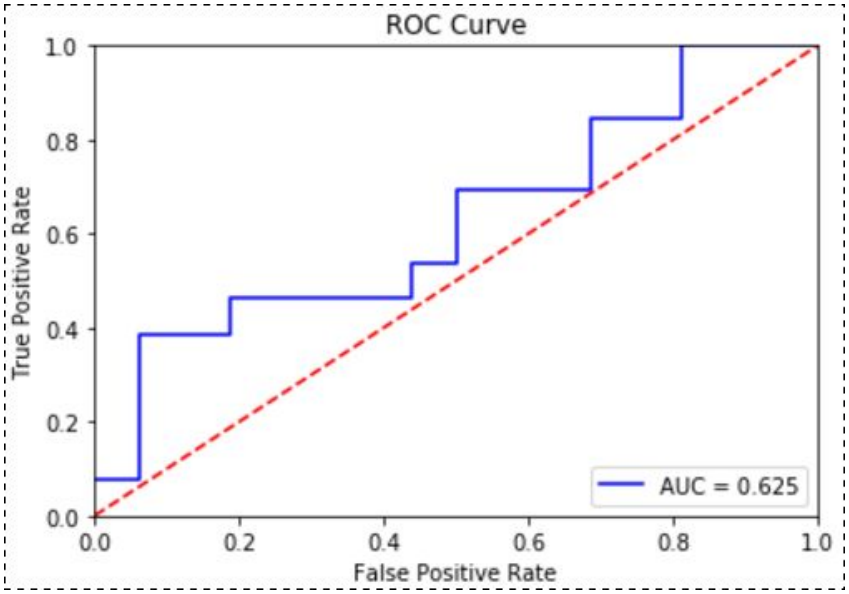
Congrats! You read 52 books of your goal of 52!

100%

```
df = pd.read_csv("df.csv")
df.head(10)
```

	Liked	Average Rating	Textbook	Pages Per Day	Year Published
0	0	2.27	0	9.090909	2016.0
1	0	2.33	1	81.000000	2015.0
2	0	2.88	0	4.945455	2014.0
3	0	2.94	0	260.000000	2014.0
4	0	3.18	0	84.000000	2009.0
5	0	3.26	0	160.000000	2017.0
6	0	3.14	1	16.000000	2015.0
7	0	3.75	0	318.000000	2008.0
8	0	2.54	0	8.470588	2016.0
9	0	2.95	0	86.000000	2016.0

```
# Model 1 (garbage… on purpose)

X = df[['Textbook', 'Pages Per Day', 'Year Published']]
y = df['Liked']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.8, random_state=0)

from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier()
model.fit(X_train, y_train)
```



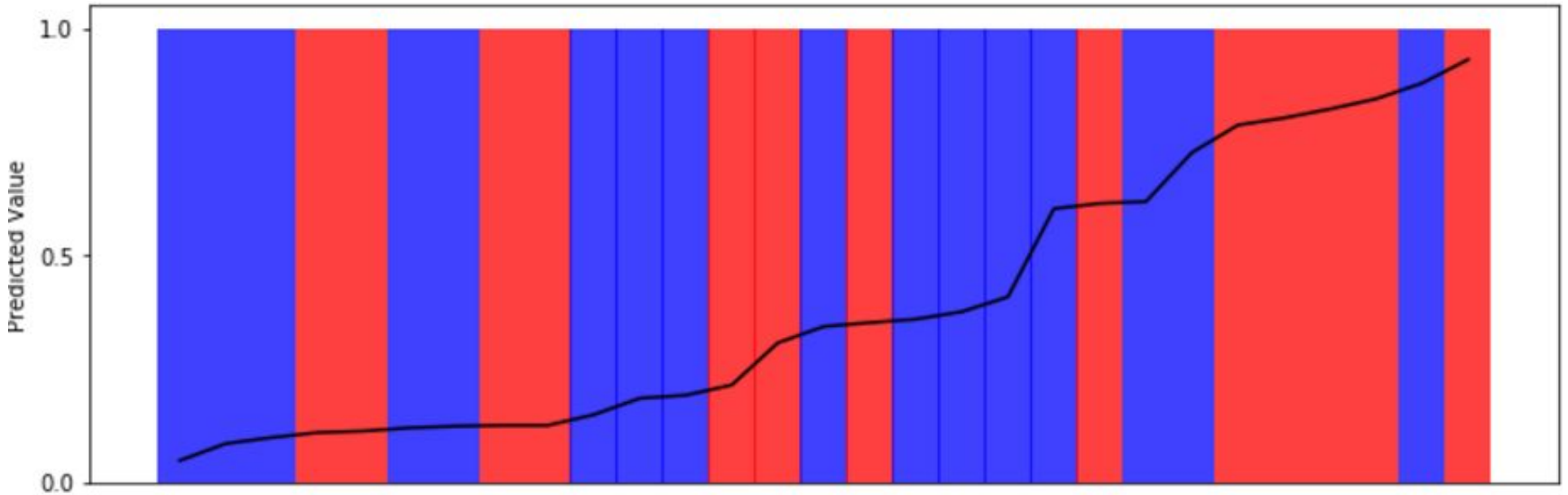

```
from sklearn.metrics import roc_curve, auc

probs = model.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(
     y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

plt.plot(fpr, tpr, 'b', label = 'AUC = {}'
     .format(roc_auc))
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve')
plt.show();
```

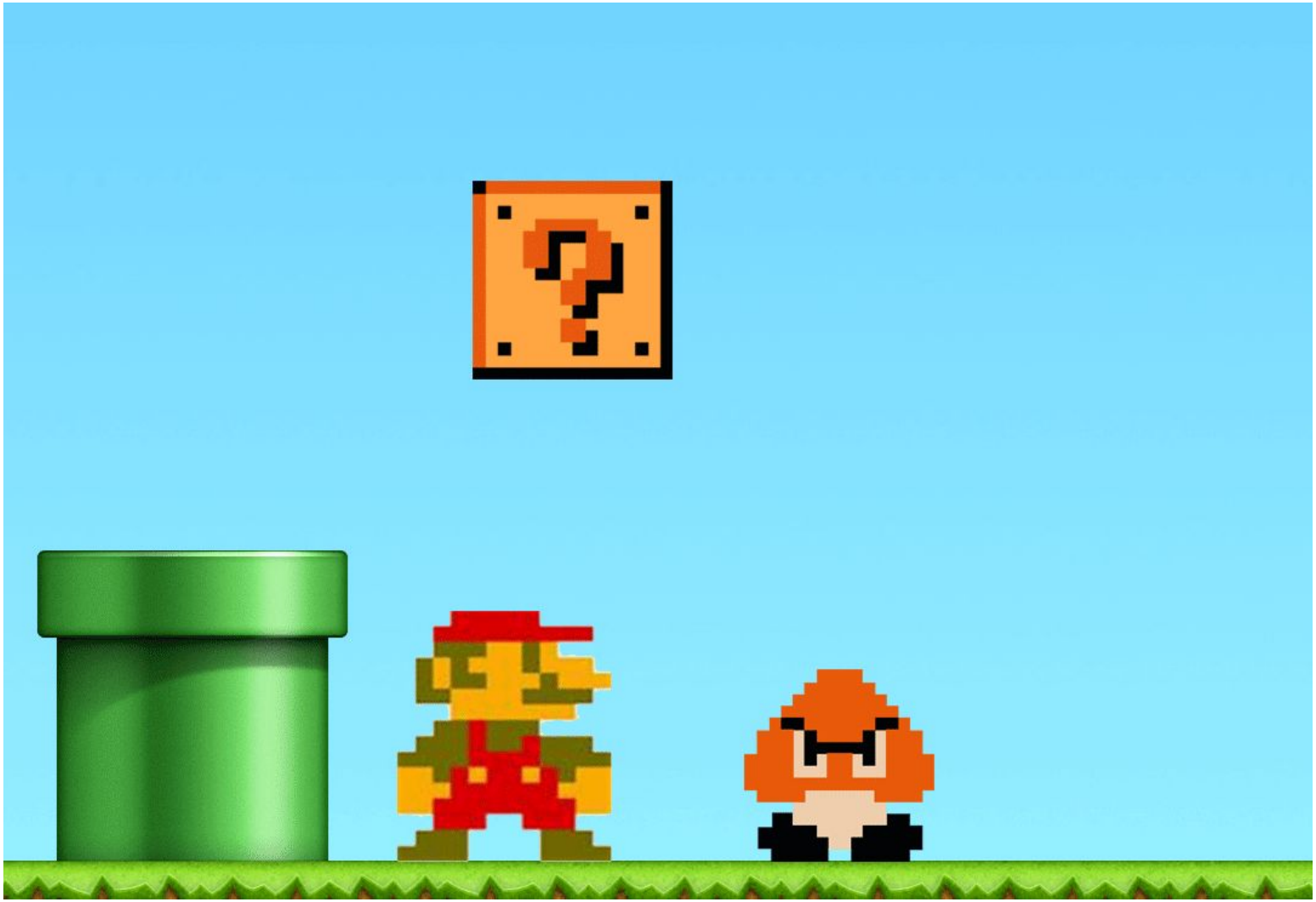
… a visual method for assessing the predictive power of models with binary outcomes. This technique allows the analyst to evaluate model fit based upon the models' ability to consistently match high-probability predictions to actual occurrences of the event of interest, and low-probability predictions to nonoccurrences of the event of interest. Unlike existing methods for assessing predictive power for logit and probit models such as Percent Correctly Predicted statistics, Brier scores, and the ROC plot, our "separation plot" has the advantage of producing a visual display that is informative and easy to explain to a general audience, while also remaining insensitive to the often arbitrary probability thresholds that are used to distinguish between predicted events and nonevents.

Source: https://scholars.duke.edu/display/pub998145

```
def separation_plot(y_true, y_pred):

    # prepare data
    sp = pd.DataFrame({'y_true': y_true, 'y_pred': y_pred})
    sp.sort_values('y_pred', inplace=True)
    sp.reset_index(level=0, inplace=True)
    sp['index'] = sp.index
    sp['height'] = 1
    sp['y_true'] = sp.y_true.astype(np.int64)
    sp['color'] = ['b' if i == 0 else 'r' for i in sp['y_true']]

    # plot data
    plt.bar(sp['index'], sp['height'], color=sp['color'],
        alpha = 0.75, width = 1.01, antialiased=True)
    plt.plot(sp['index'], sp['y_pred'], c='black')
    plt.xticks([])
    plt.yticks([0, 0.5, 1])
    plt.ylabel('Predicted Value')
    plt.show()
```

```
y_true = y_test
y_pred = model.predict_proba(X_test)[:, 1]
separation_plot(y_true, y_pred)
```

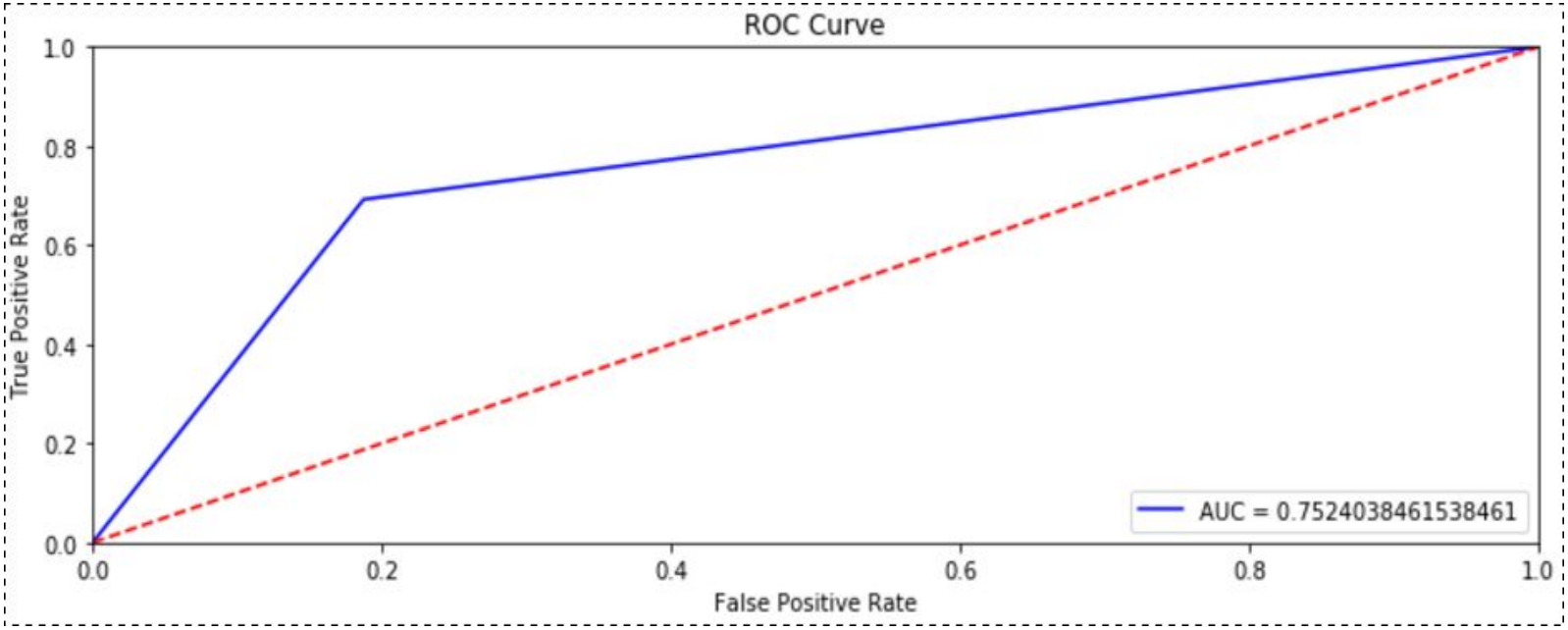




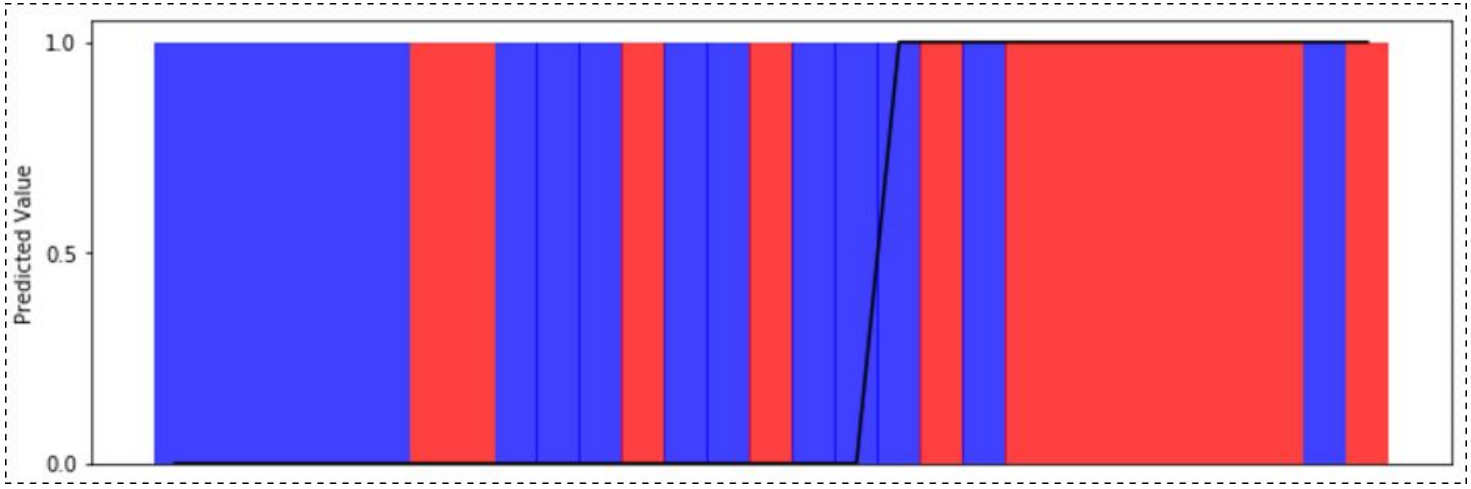
Animated GIF

```
X = df[['Average Rating', 'Pages Per Day']]
y = df['Liked']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.8, random_state=0)

from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(X_train, y_train)
```

```
plt.title('ROC Curve')
plt.plot(fpr, tpr, 'b', label = 'AUC =
{}'.format(roc_auc))
plt.legend(loc = 'lower right')
plt.show();
```

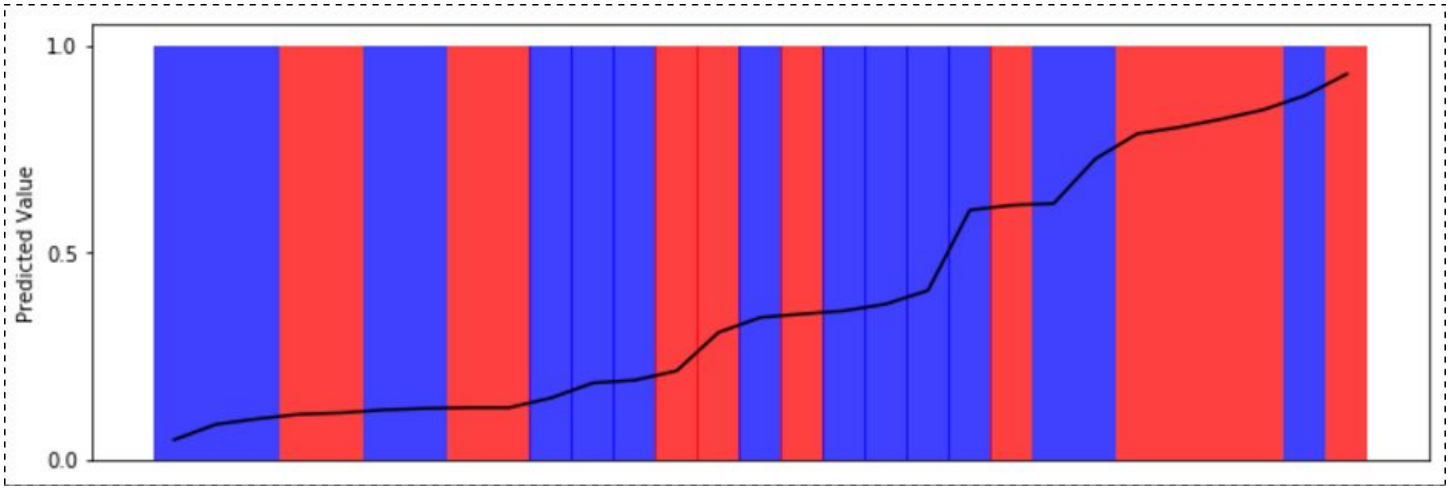


```
y_true = y_test
y_pred = model.predict_proba(X_test)[:, 1]
separation_plot(y_true, y_pred)
```

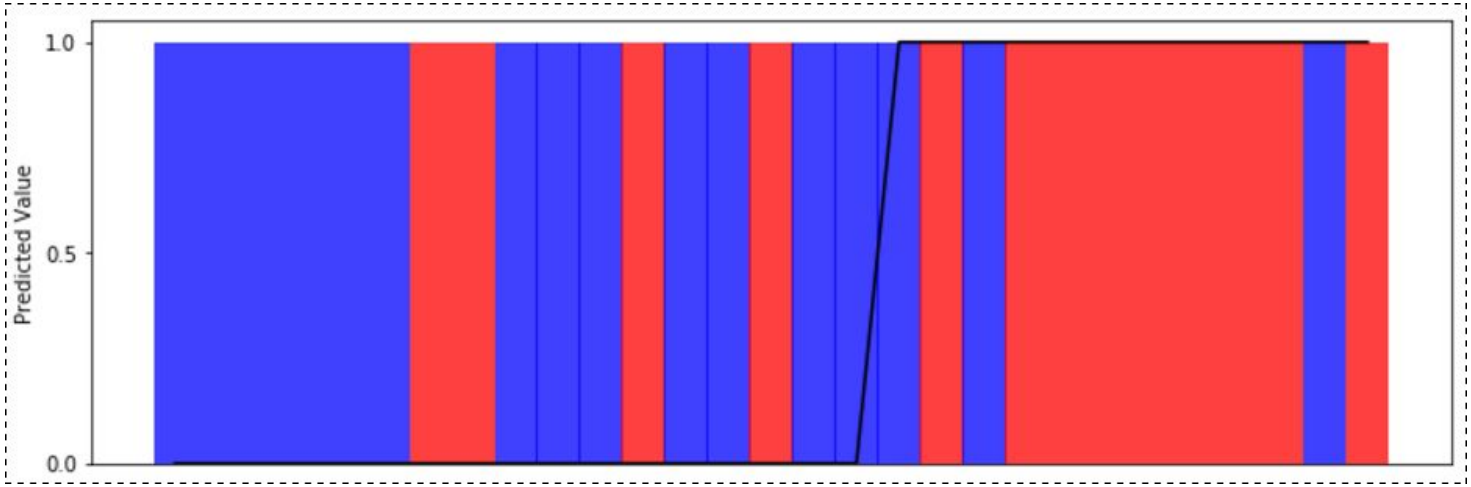


```
y_true = y_test
y_pred = model.predict_proba(X_test)[:, 1]
separation_plot(y_true, y_pred)
```

#1

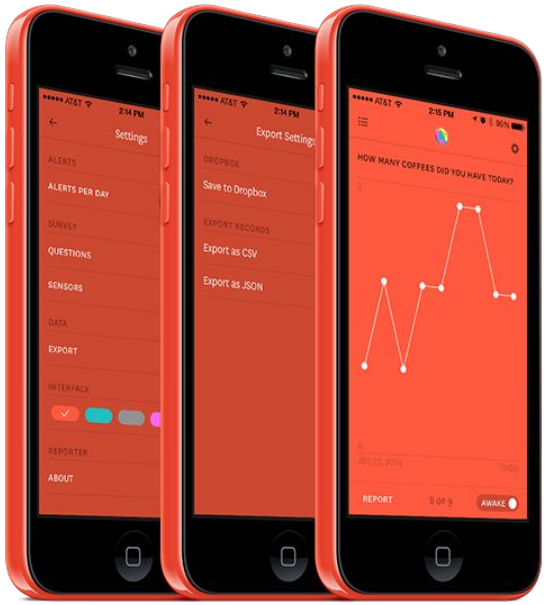


#2



3/3

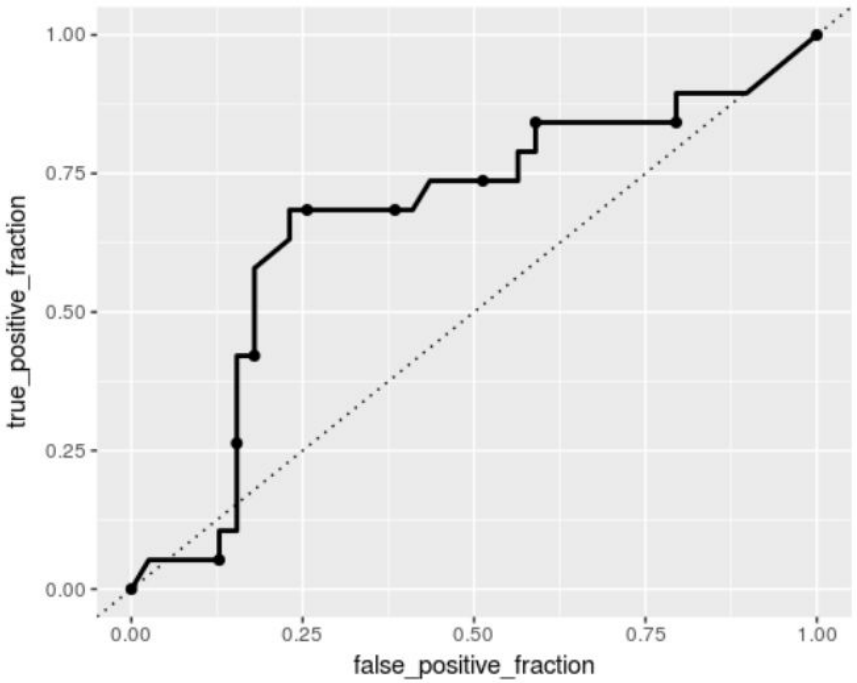
Animated GIF


Settings
ALERTS
ALERTS PER DAY
SURVEY
QUESTIONS
SENSORS
DATA
EXPORT
INTERFACE
REPORTER
ABOUT
Export Settings
DROPBOX
Save to Dropbox
EXPORT RECORDS
Export as CSV
Export as JSON
HOW MANY COFFEES DID YOU HAVE TODAY?
REPORT
AWAKE

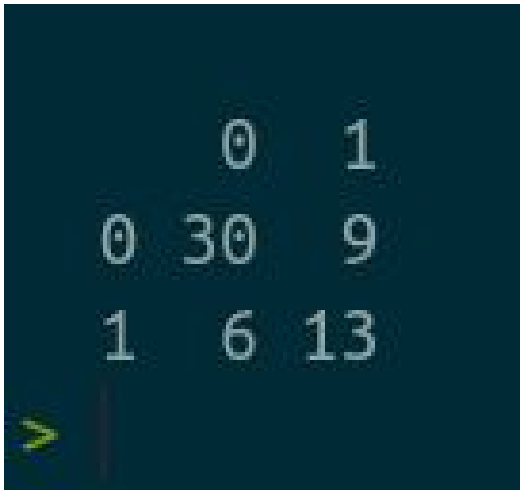
log	happy	who	what	where	weather
265	0	Acquaintance	Eating	USA	Rain
266	0	Alone	Resting	USA	Cloudy
267	0	Alone	Commuting	USA	Clear
268	0	Alone	Commuting	USA	Clear
269	0	Alone	Commuting	Waterloo	Overcast
270	0	Alone	Commuting	Waterloo	Overcast
271	1	Girlfriend	Partying	London	Cloudy
272	1	Girlfriend	Recovering	London	Overcast
273	1	Girlfriend	Resting	London	Overcast
274	1	Girlfriend	Eating	London	Overcast
275	1	Girlfriend	Playing	London	Cloudy
276	0	Alone	Listening	Travel	Overcast
277	1	Alone	Making	Home	Overcast
278	0	Alone	Resting	Home	Snow
279	1	Friend	Making	Waterloo	Snow
280	1	Friend	Partying	Waterloo	Snow
281	0	Alone	Resting	Home	Overcast

```
library(tidyverse)

df <- read_csv("df.csv") %>% select(-log)

TI <- caret::createDataPartition(
    y=df$happy, p=0.80, list=FALSE)

train <- df[TI, ]
test <- df[-TI, ]

mod <- glm(happy ~ ., data=train, family='binomial')
summary(mod)

test$pred <- predict(mod, test, 'response')
```

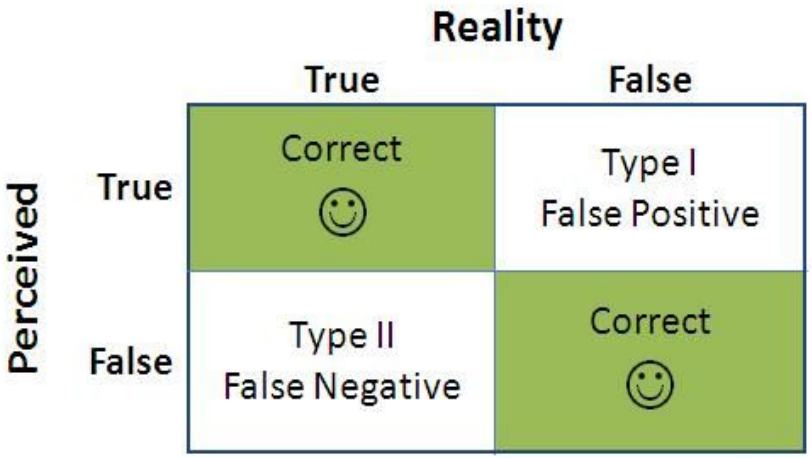
	Estimate
(Intercept)	-2.1132
whoAlone	-0.7305
whoCoworker	1.1494
whoFamily	-0.7991
whoFriend	3.2789
whoGirlfriend	4.8521
whatCommuting	-1.3861
whatEating	-16.4230
whatExercising	23.6069
whatGaming	20.5820
whatListening	19.4858
whatMaking	21.8877
whatOther-ing	15.4249
whatPartying	19.7572
whatPlaying	19.3237
whatReading	-28.6820
whatRecovering	-1.2094
whatResting	17.7610
whatShopping	2.3380
whatShowering	39.2955




```
library(plotROC)

p <- ggplot(test, aes(
    d = happy, m = pred)) +
geom_roc(labels=FALSE) +
geom_abline(slope=1, lty=3)

calc_auc(p)$AUC

[1] 0.70
```

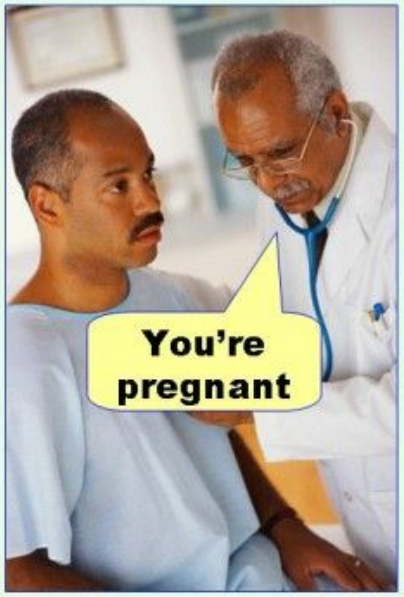



```
test$pred <- predict(
    mod, test, type="response")
test$pred <- ifelse(
    test$pred >= 0.5, 1, 0)
table(test$happy, test$pred)
```

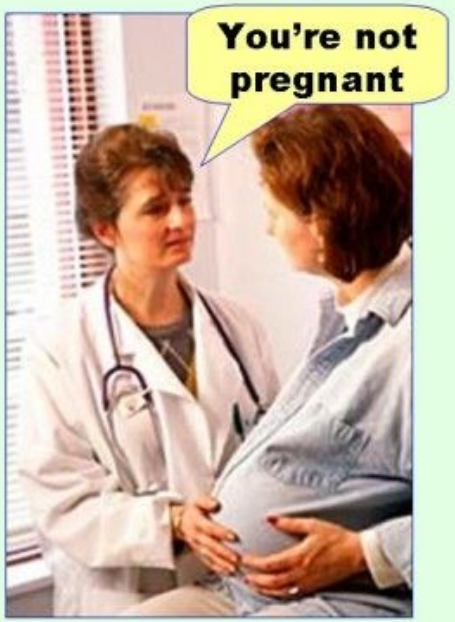



```
test$pred <- predict(
    mod, test, type="response")
test$pred <- ifelse(
    test$pred >= 0.5, 1, 0)
table(test$happy, test$pred)
```

Type I error
(false positive)

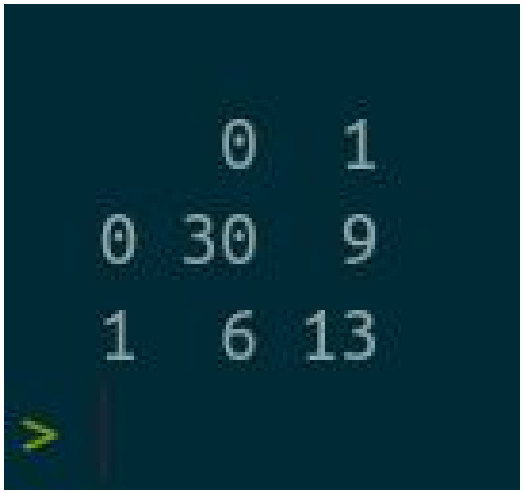


Type II error
(false negative)




```
ct(
="response")
e(
 5, 1, 0)
test$pred)
```

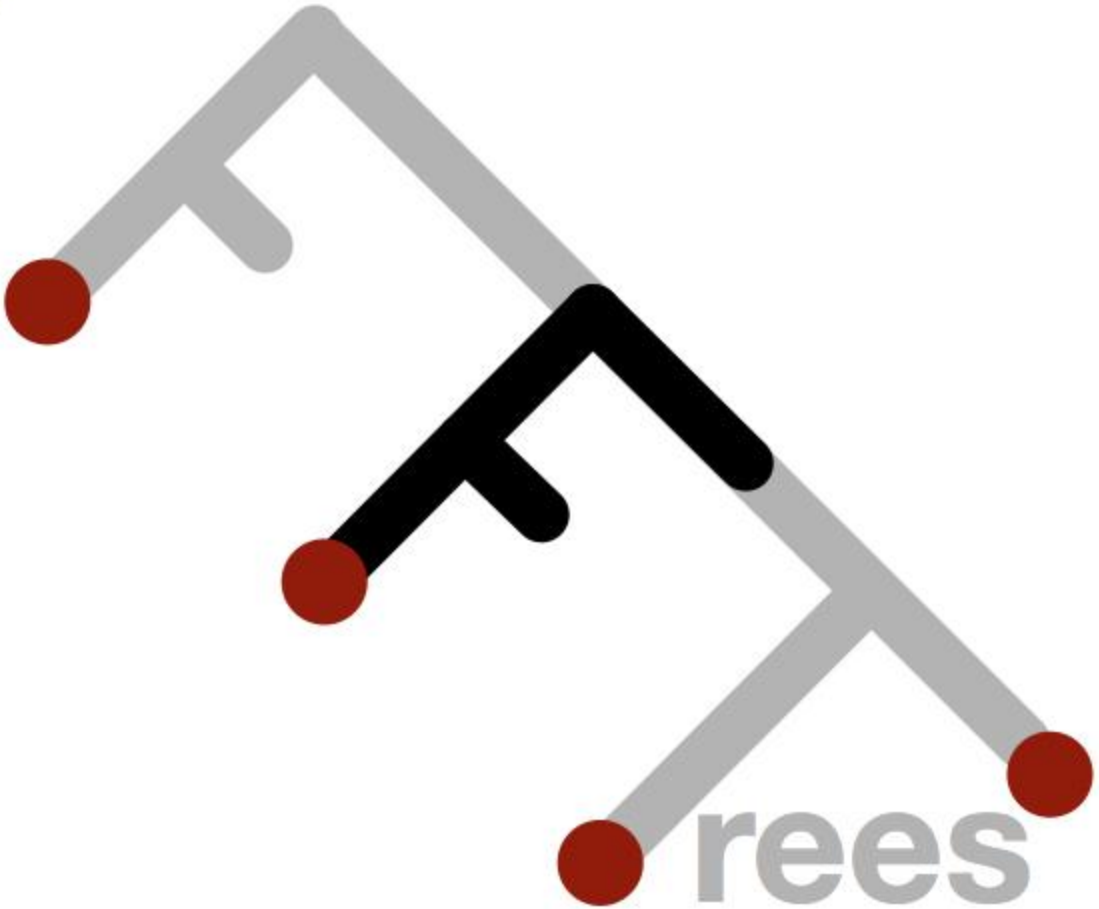
Animated GIF




```
table(test$happy, test$pred) %>%
  as_data_frame() %>%
  rename(truth=Var1, decision=Var2) %>%
  mutate(truth=ifelse(truth==1,
    "Happy", "Not Happy")) %>%
  mutate(decision=ifelse(decision==1,
    "Happy", "Not Happy")) %>%
  ggplot(aes(x = truth, y = decision)) +
  geom_point(aes(shape=decision,
    color=truth, size=n)) +
  geom_text(aes(label = n)) +
  scale_size_continuous(
    range = c(5, 20)) +
  scale_color_manual(
    values = c("green", "red"))
```

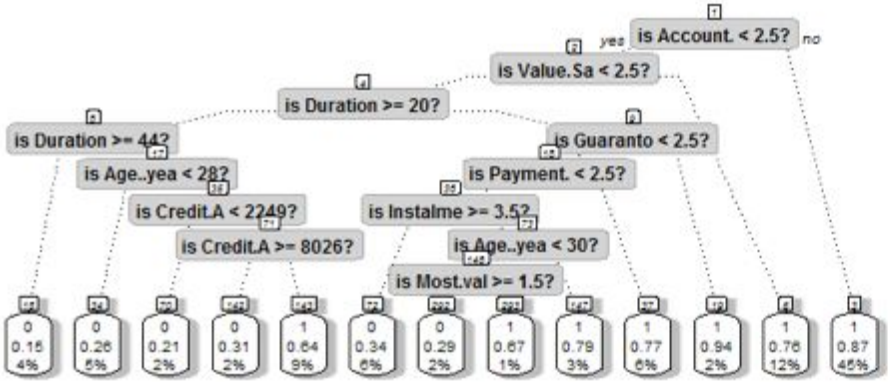

```
table(test$happy, test$pred) %>%
  as_data_frame() %>%
  rename(truth=Var1, decision=Var2) %>%
  mutate(truth=ifelse(truth==1,
    "Happy", "Not Happy")) %>%
  mutate(decision=ifelse(decision==1,
    "Happy", "Not Happy")) %>%
  ggplot(aes(x = truth, y = decision)) +
  geom_point(aes(shape=decision,
    color=truth, size=n)) +
  geom_text(aes(label = n)) +
  scale_size_continuous(
    range = c(5, 20)) +
  scale_color_manual(
    values = c("green", "red"))
```

Animated GIF

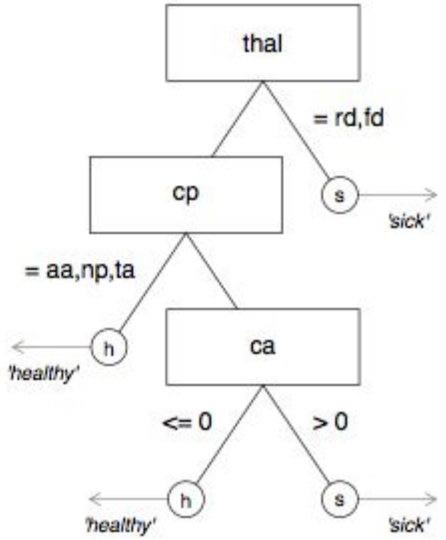


https://github.com/ndphillips/

How can people make good decisions based on limited, noisy information?... Fast-and-frugal decision trees (FFT) were developed by Green & Mehr (1997). An FFT is a decision tree with exactly two branches from each node, where one, or both, of the branches are exit branches (Martignon et al., 2008). FFTrees are transparent, easy to modify, and accepted by physicians (unlike regression).

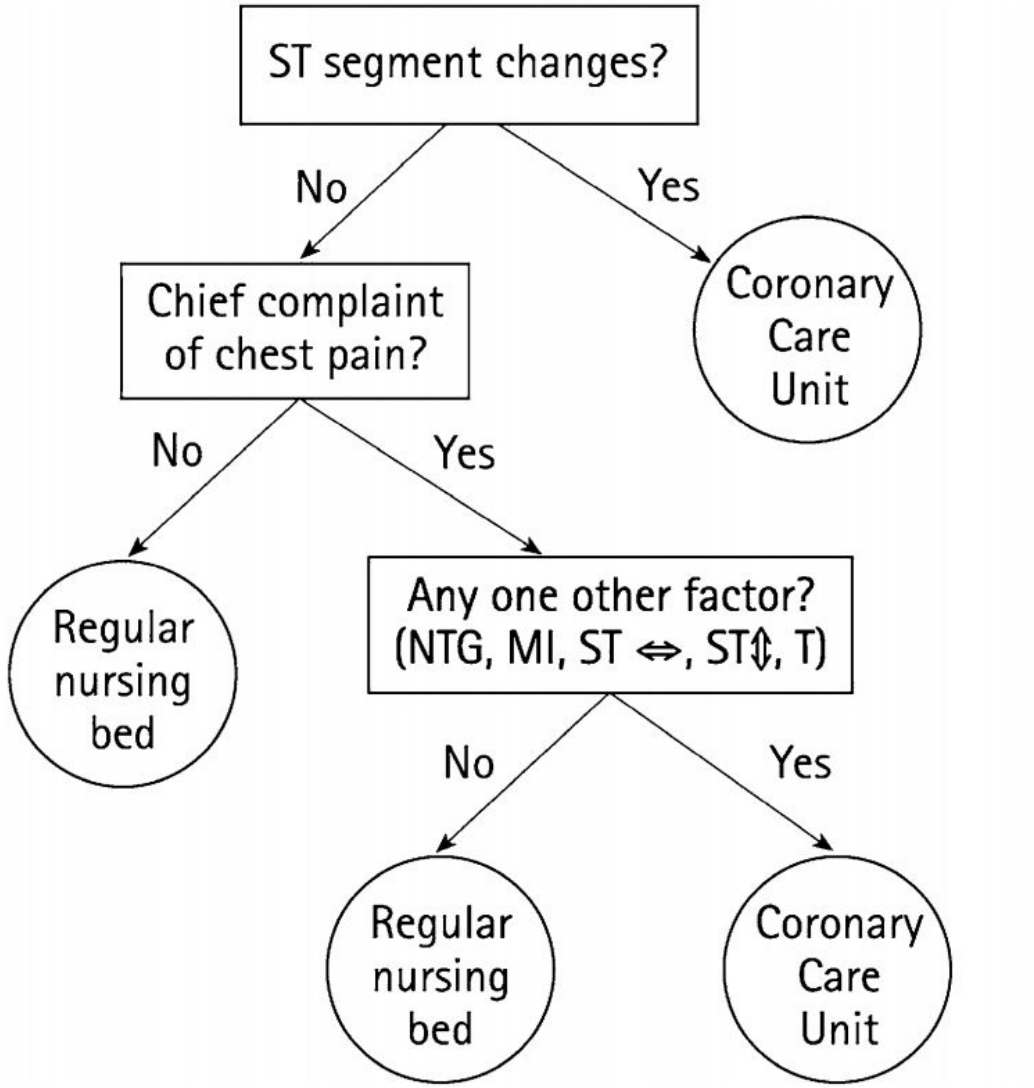


VS



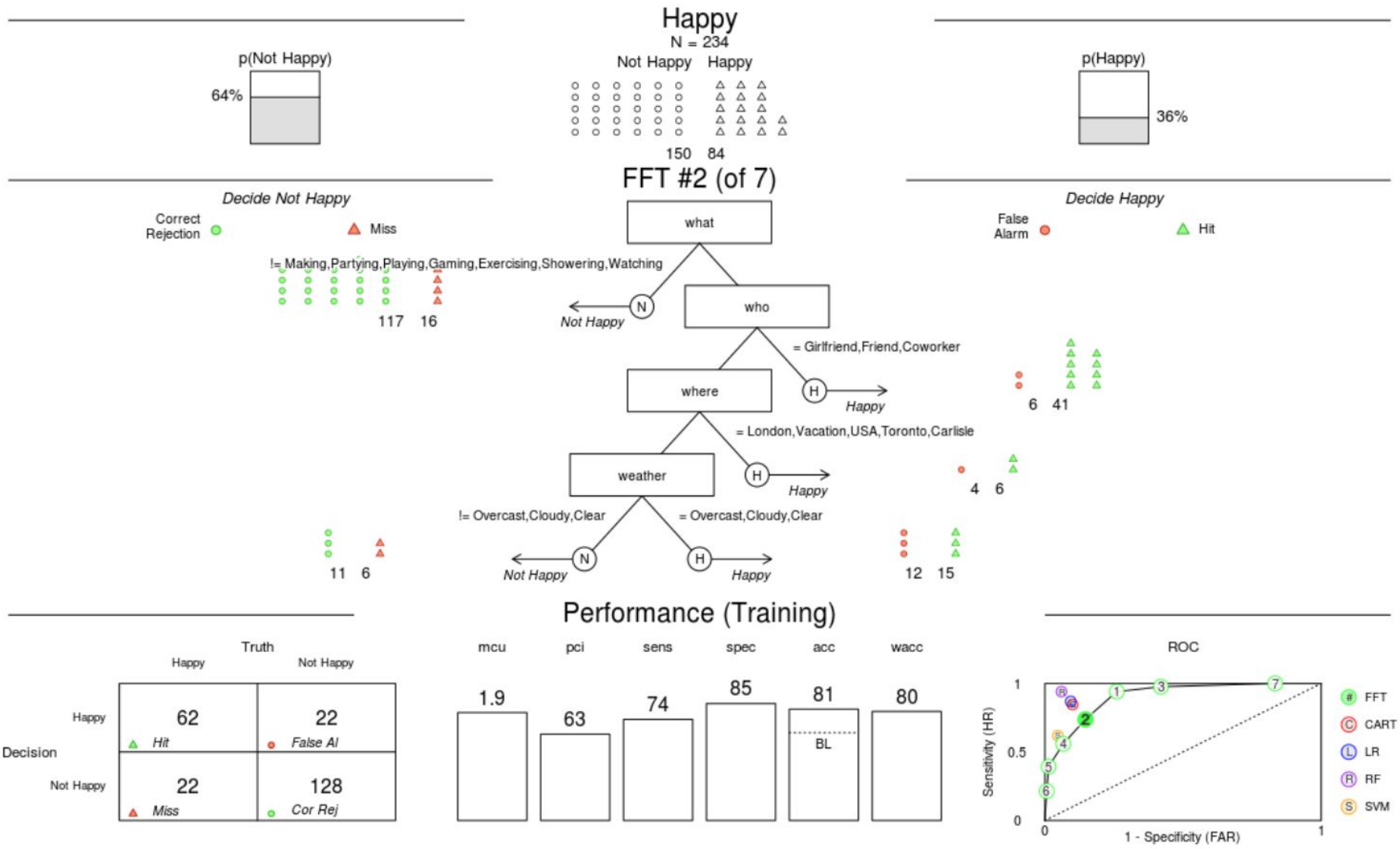
Standard

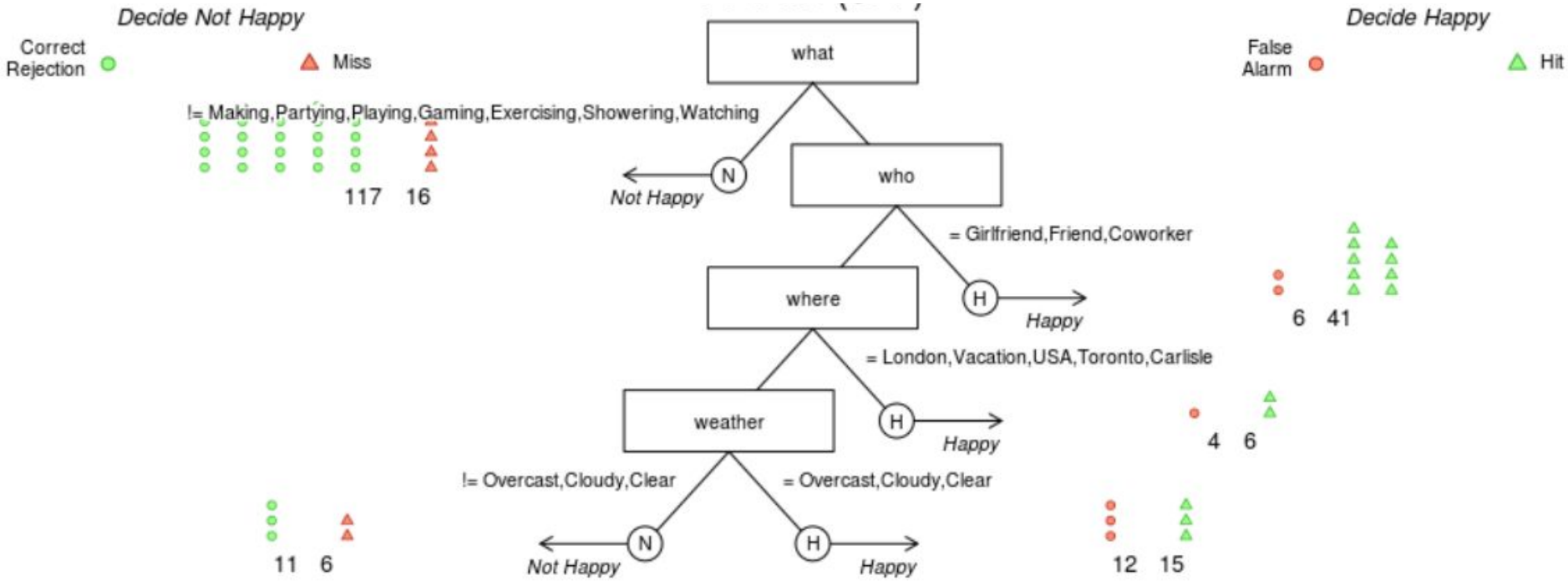
Fast-and-Frugal

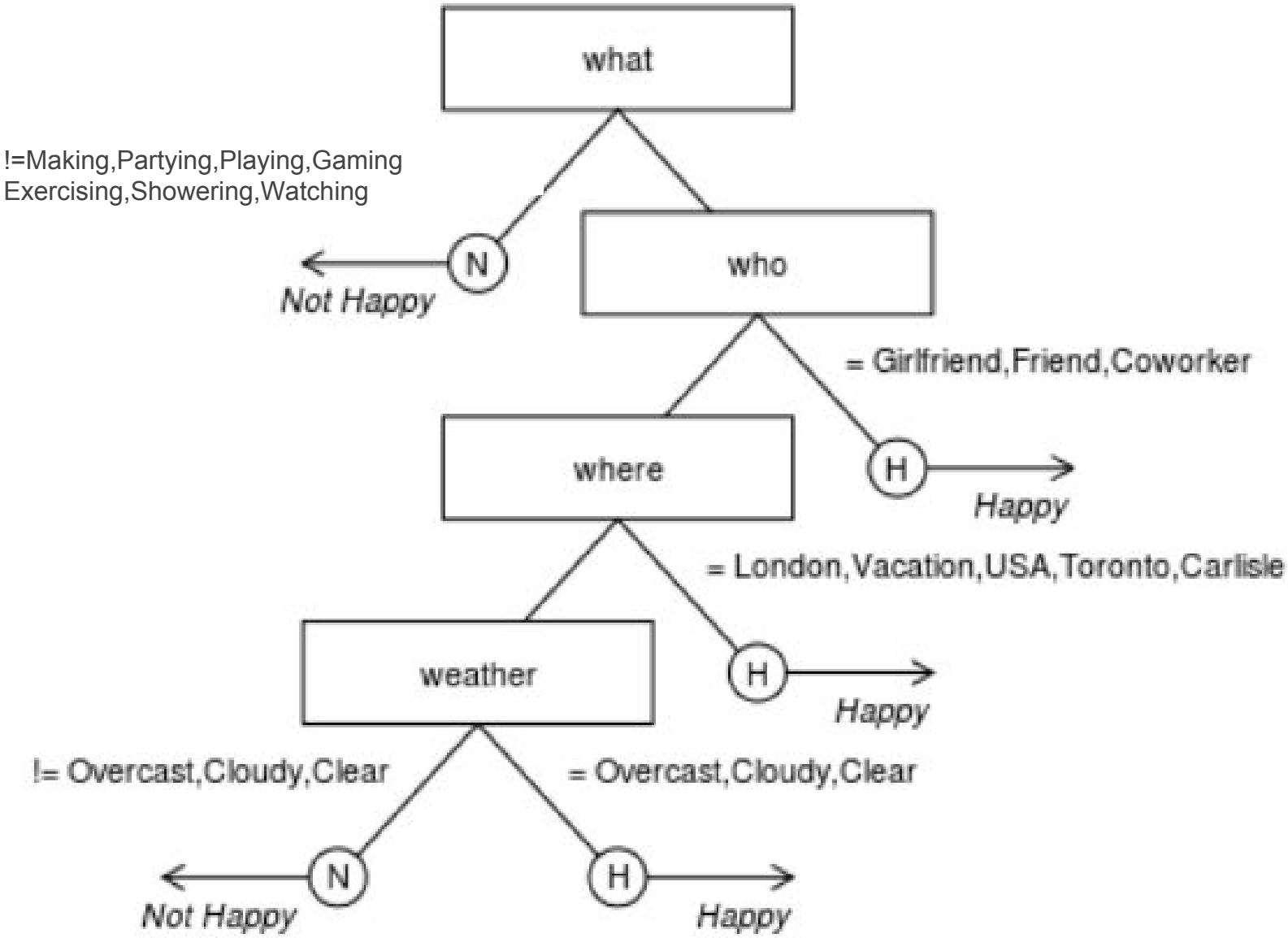

ST segment changes?
No
Yes
Chief complaint of chest pain?
Coronary Care Unit
No
Yes
Regular nursing bed
Any one other factor? (NTG, MI, ST ⇔, ST⇕, T)
No
Yes
Regular nursing bed
Coronary Care Unit

```
# install.packages("FFTrees")
library(FFTrees)

fft <- FFTrees(happy ~., data = train, main = "Happy",
    decision.labels = c("Not Happy", "Happy"))

plot(fft)
```

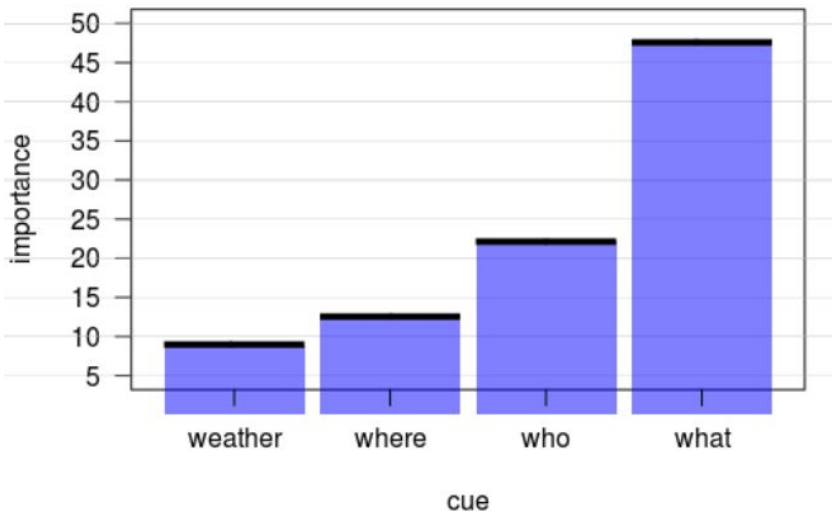
```
plot(fft,tree=2)
```




Decide Not Happy
Correct Rejection
Miss
Decide Happy
False Alarm
Hit
what
!= Making,Partying,Playing,Gaming,Exercising,Showering,Watching
N
Not Happy
who
= Girlfriend,Friend,Coworker
where
H
Happy
= London,Vacation,USA,Toronto,Carlisle
weather
H
Happy
!= Overcast,Cloudy,Clear
= Overcast,Cloudy,Clear
N
Not Happy
H
Happy
117 16
6 41
4 6
11 6
12 15


what
!=Making,Partying,Playing,Gaming
Exercising,Showering,Watching
N
Not Happy
who
= Girlfriend,Friend,Coworker
where
H
Happy
= London,Vacation,USA,Toronto,Carlisle
weather
H
Happy
!= Overcast,Cloudy,Clear
= Overcast,Cloudy,Clear
N
Not Happy
H
Happy

```
> inwords(fft)

$v1
[1] "If what =
{Making,Partying,Playing,Gaming,Exercising,Showering,Watching},
predict Happy"
[2] "If who != {Girlfriend,Friend,Coworker}, predict Not Happy"
[3] "If where != {London,Vacation,USA,Toronto,Carlisle}, predict
Not Happy, otherwise, predict Happy"

$v2
[1] "If what =
{Making,Partying,Playing,Gaming,Exercising,Showering,Watching},
predict Happy. If who != {Girlfriend,Friend,Coworker}, predict
Not Happy. If where != {London,Vacation,USA,Toronto,Carlisle},
predict Not Happy, otherwise, predict Happy"
```

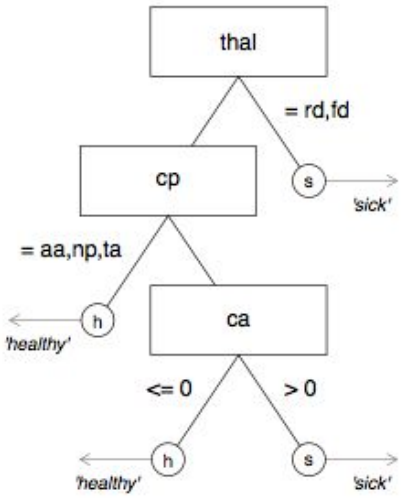


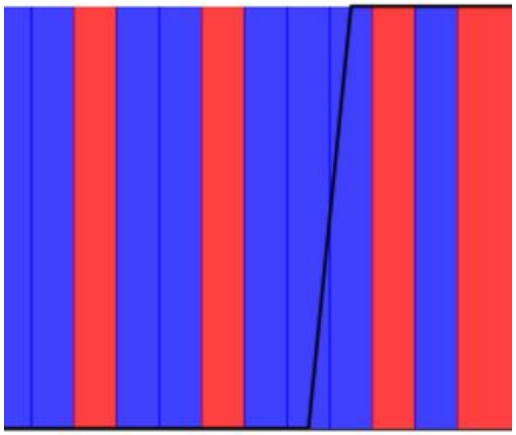

```
importance <- fft$comp$rf$model$importance

importance <- data.frame(
     cue =
rownames(fft$comp$rf$model$importance),
     importance = importance[,1])

importance <-
importance[order(importance$importance),]
```

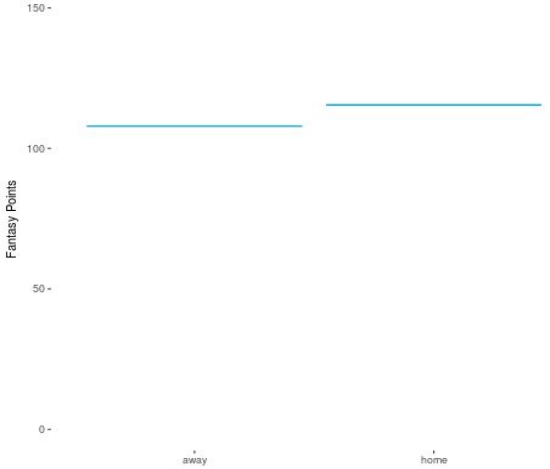
summary

Animated GIF

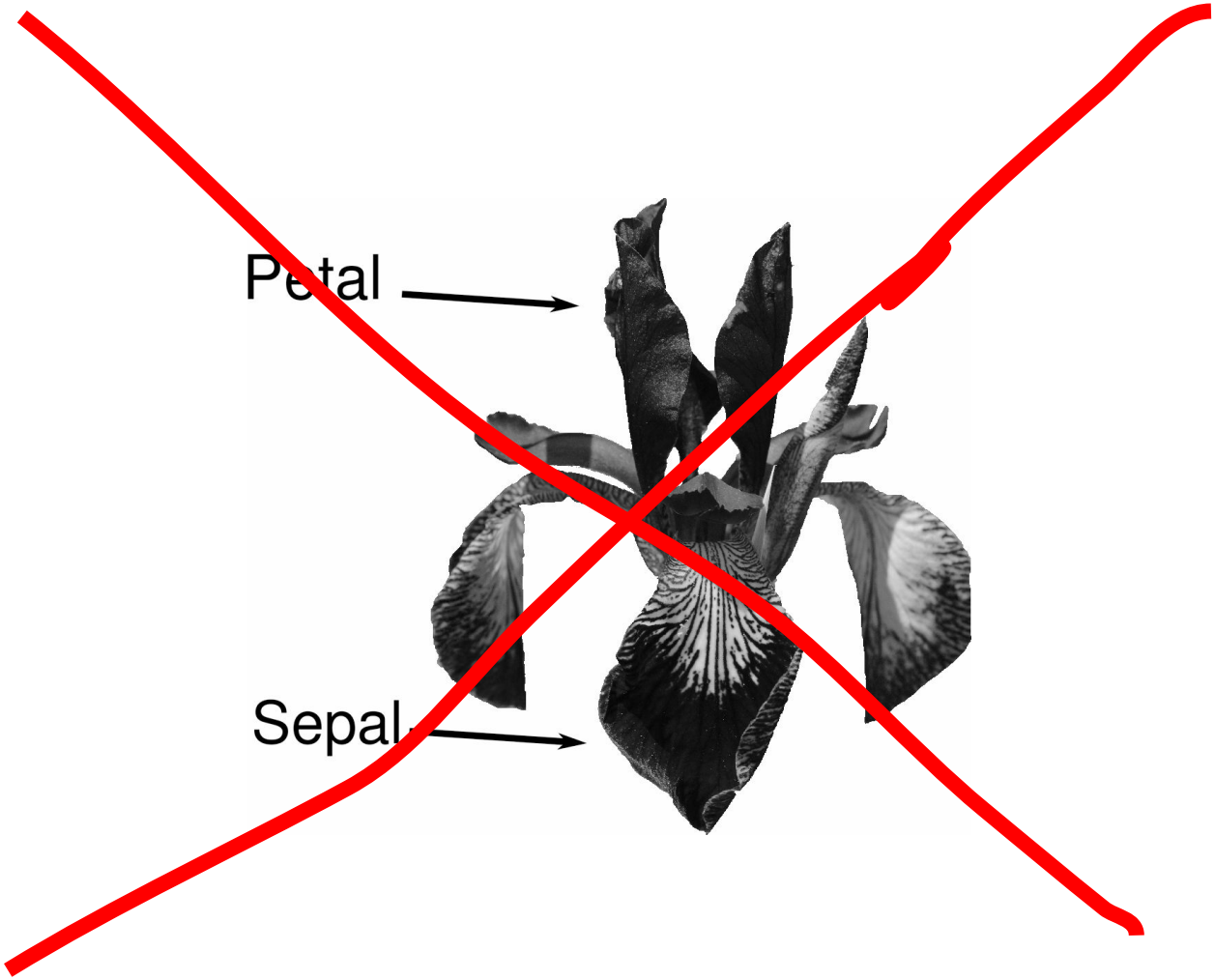
Binary Data







Simulation


Petal
Sepal























#ODSC

maxhumber