

Jake's pups!





**Lalita's photography!**



# Logistic regression

## Data Science for Biologists

---

Dr. Spielman

# Make it fit better on slides

```
penguins %>%
  rename(bill_len = bill_length_mm,
         bill_dep = bill_depth_mm,
         flipper = flipper_length_mm,
         mass = body_mass_g) -> peng
```

peng

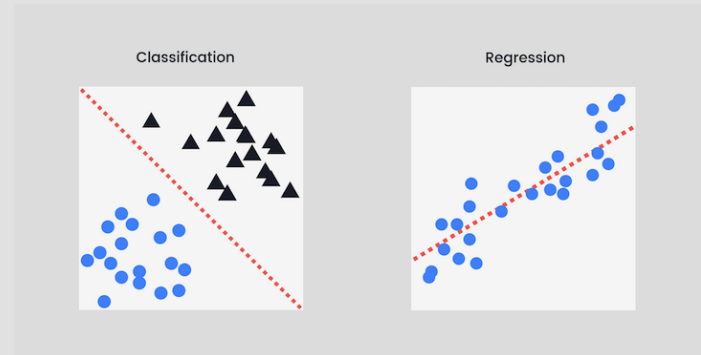
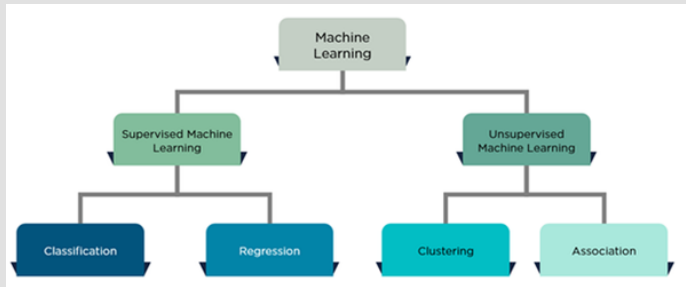
```
## # A tibble: 344 x 8
##   species island bill_len bill_dep flipper mass sex
##   <fct>   <fct>   <dbl>   <dbl>   <int> <int> <fct>
## 1 Adelie  Torge...   39.1    18.7    181  3750 male
## 2 Adelie  Torge...   39.5    17.4    186  3800 fema...
## 3 Adelie  Torge...   40.3     18    195  3250 fema...
## 4 Adelie  Torge...    NA     NA     NA    NA <NA>
## 5 Adelie  Torge...   36.7    19.3    193  3450 fema...
## 6 Adelie  Torge...   39.3    20.6    190  3650 male
## 7 Adelie  Torge...   38.9    17.8    181  3625 fema...
## 8 Adelie  Torge...   39.2    19.6    195  4675 male
## 9 Adelie  Torge...   34.1    18.1    193  3475 <NA>
## 10 Adelie Torge...   42     20.2    190  4250 <NA>
## # ... with 334 more rows, and 1 more variable:
## #   year <int>
```

# Linear regression vs. logistic regression

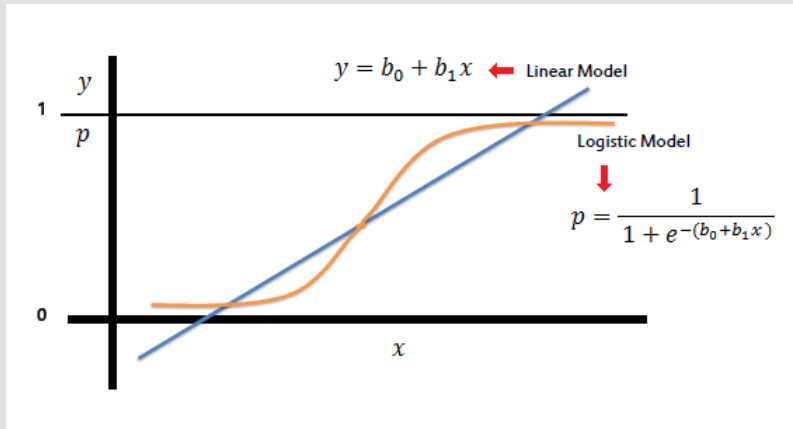
- Linear regression: How much do these (linearly-related) predictors explain variation in my *numeric* response variable?
- Logistic regression: How well do these predictors explain variation in my *categorical **binary*** response variable?
  - E.g. predicting Species in the iris dataset would be a categorical predictor, but NOT binary
  - Type of classifier

# Where are we in the "machine learning" universe?

- Machine learning = the computer learns through experience
  - More data = more experience! *Training models on data IS machine learning*
  - Ignore the AI hype.



# Logistic regression



- Linear regression:  $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 \dots + \beta_N X_N + \epsilon$
- Logistic regression *transforms the predictors*
  - $t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 \dots + \beta_N X_N + \epsilon$
  - $Y = \frac{1}{1 + e^{-t}}$  (or,  $p = \dots$  in image)

```
library(palmerpenguins) # to access penguins dataset
```

```
## Make it fit better on slides
```

```
penguins%>%  
  rename(bill_len = bill_length_mm,  
         bill_dep = bill_depth_mm,  
         flipper = flipper_length_mm,  
         mass     = body_mass_g) %>%  
  # appease the step() function gods  
  drop_na()-> peng_nona
```

```
peng_nona
```

```
## # A tibble: 333 x 8  
##   species island bill_len bill_dep flipper mass sex  
##   <fct>   <fct>   <dbl>   <dbl>   <int> <int> <fct>  
## 1 Adelie  Torge...   39.1    18.7    181  3750 male  
## 2 Adelie  Torge...   39.5    17.4    186  3800 fema...  
## 3 Adelie  Torge...   40.3     18    195  3250 fema...  
## 4 Adelie  Torge...   36.7    19.3    193  3450 fema...  
## 5 Adelie  Torge...   39.3    20.6    190  3650 male  
## 6 Adelie  Torge...   38.9    17.8    181  3625 fema...  
## 7 Adelie  Torge...   39.2    19.6    195  4675 male  
## 8 Adelie  Torge...   41.1    17.6    182  3200 fema...  
## 9 Adelie  Torge...   38.6    21.2    191  3800 male  
## 10 Adelie Torge...   34.6    21.1    198  4400 male  
## # ... with 323 more rows, and 1 more variable:  
## #   year <int>
```



# Goal: Can we predict a penguin's sex?

```
## Sex is a binary variable in the penguins data:  
levels(peng_nona$sex)
```

```
## [1] "female" "male"
```

## Step 1: Prepare the data

```
## To appease the R gods, and make your life easier, ...  
## re-code the response as arbitrary success = 1, failure = 0.  
peng_nona %>%  
  # change `sex` to be 0, 1  
  mutate(sex = if_else(sex == "female", 1, 0)) -> peng_nona_sex01  
  
peng_nona_sex01
```

```
## # A tibble: 333 x 8  
##   species island bill_len bill_dep flipper mass sex  
##   <fct>    <fct>    <dbl>    <dbl>   <int> <int> <dbl>  
## 1 Adelie  Torge...    39.1    18.7    181  3750     0  
## 2 Adelie  Torge...    39.5    17.4    186  3800     1  
## 3 Adelie  Torge...    40.3     18    195  3250     1  
## 4 Adelie  Torge...    36.7    19.3    193  3450     1  
## 5 Adelie  Torge...    39.3    20.6    190  3650     0
```

## Step 2: Build the model

```
glm(response ~ predictors, data = data, family = "binomial")
```

- Use function `glm()`
- Include argument `family = "binomial"`
- Everything else is the same!

```
## Use model selection to identify optimal predictors  
initial_fit <- glm(sex ~ ., data = peng_nona_sex01, family = "binomial")  
fit <- step(initial_fit, trace = FALSE)
```

# Interpreting the logistic regression coefficients

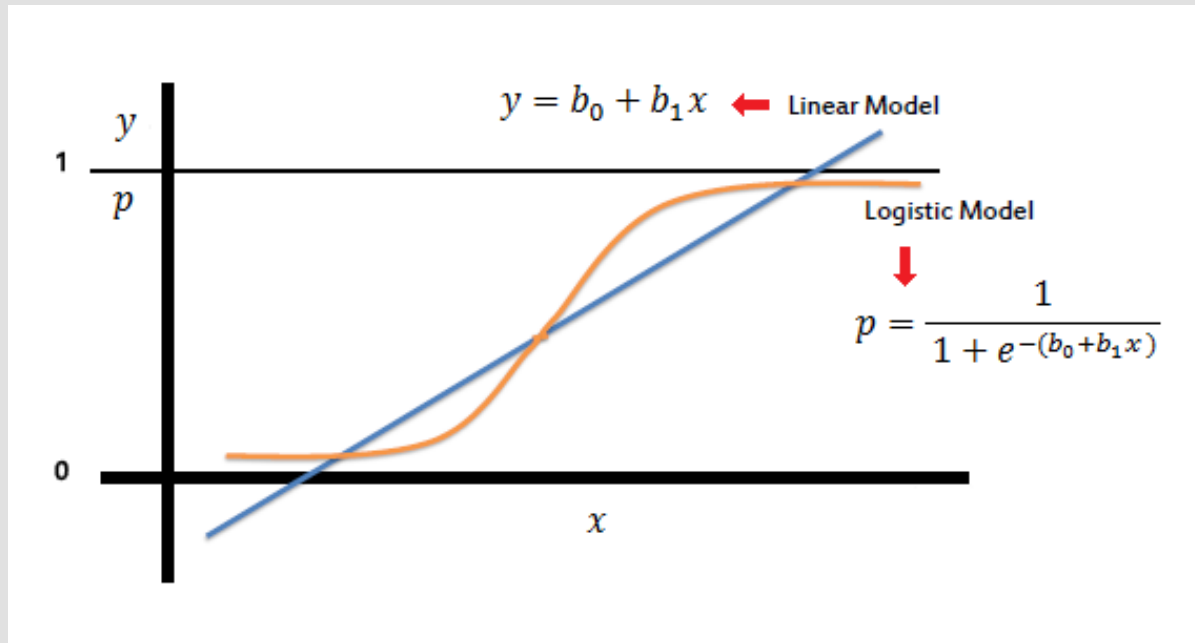
Proip: Don't.

```
tidy(fit)
```

```
## # A tibble: 6 x 5
##   term                estimate std.error statistic  p.value
##   <chr>              <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)       76.1      9.80      7.76 8.34e-15
## 2 speciesChinstr...  6.90     1.56      4.43 9.56e- 6
## 3 speciesGentoo     7.88     2.25      3.50 4.62e- 4
## 4 bill_len          -0.620    0.131     -4.74 2.11e- 6
## 5 bill_dep          -1.64     0.331     -4.97 6.69e- 7
## 6 mass              -0.00596 0.00106    -5.62 1.88e- 8
```

- For every unit increase in the predictor, the **log odds of success** of the response increases by the coefficient
  - $Pr(success)$  = probability of being *female* for a given set of observations (predictors)
  - $Pr(failure)$  = probability of being *male* for a given set of observations
  - **Log odds** =  $ln\left(\frac{Pr(success)}{Pr(failure)}\right)$

# Visualizing the fitted logistic curve ("the model")



```
## USING head() to make it fit on slides!!
```

```
## YOUR X-AXIS !!
```

```
## What would have been your Y-values if this were regression
```

```
head(fit$linear.predictors)
```

```
##           1           2           3           4           5
## -1.2431192  0.3474612  2.1441466  1.0474121 -3.8938334
##           6
##  1.1053579
```

```
## YOUR Y-AXIS !!
```

```
## The logit transformed - PROBABILITIES OF SUCCESS
```

```
head(fit$fitted.values)
```

```
##           1           2           3           4           5
## 0.22389350 0.58600180 0.89512053 0.74027765 0.01996058
##           6
## 0.75126267
```

- $t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 \dots + \beta_N X_N + \epsilon$
- $Y = \frac{1}{1+e^{-t}}$

```
1/(1 + exp(-1 * fit$linear.predictors)) %>% head()
```

```
##           1           2           3           4           5
## 0.22389350 0.58600180 0.89512053 0.74027765 0.01996058
##           6
```



# Visualizing the model: Prepare the data

```
tibble(x = fit$linear.predictors,  
       y = fit$fitted.values,  
       sex = peng_nona$sex) -> fit_tibble
```

```
fit_tibble
```

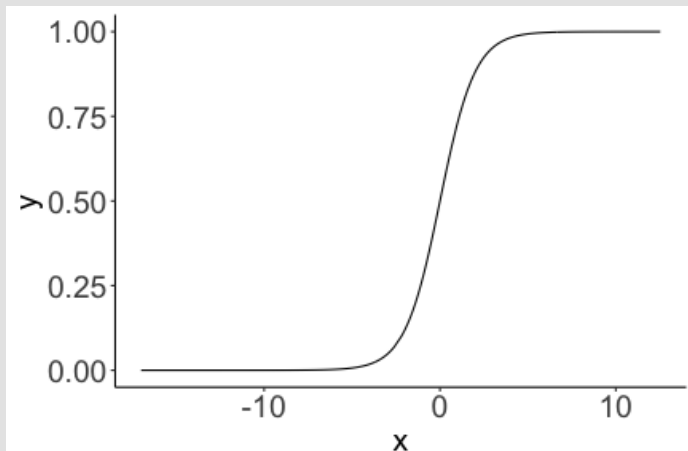
```
## # A tibble: 333 x 3  
##       x      y sex  
##   <dbl> <dbl> <fct>  
## 1 -1.24  0.224  male  
## 2  0.347 0.586  female  
## 3  2.14  0.895  female  
## 4  1.05  0.740  female  
## 5 -3.89  0.0200  male  
## 6  1.11  0.751  female  
## 7 -8.30  0.000249 male  
## 8  2.60  0.931  female  
## 9 -5.34  0.00477  male  
## 10 -6.27  0.00188  male  
## # ... with 323 more rows
```

# Visualizing the model

```
head(fit_tibble)
```

```
## # A tibble: 6 x 3
##       x         y sex
##   <dbl> <dbl> <fct>
## 1 -1.24  0.224 male
## 2  0.347 0.586 female
## 3  2.14  0.895 female
## 4  1.05  0.740 female
## 5 -3.89  0.0200 male
## 6  1.11  0.751 female
```

```
ggplot(fit_tibble, aes(x = x, y = y))
  geom_line() +
  theme(legend.position = "bottom")
```



# Visualizing the model *fully*

```
ggplot(fit_tibble, aes(x = x, y = y)) +  
  geom_line() +  
  geom_point(aes(color = sex), alpha = 0.4) +  
  scale_color_brewer(palette = "Set1") +  
  theme(legend.position = "bottom") +  
  labs(x = "Linear predictors",  
       y = "Probability the penguin is female")
```

# Confusion matrix time

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

- **First ask:** is the result positive or negative?
  - "Successes" are positive and "failures" are negative.
- **Then ask:** should we have gotten that result though?
  - If yes, *TRUE*. If not, *FALSE*.

# What is it?

A new arthritis drug does help pain clinical trials, even though it actually does reduce arthritis pain.

A person with HIV receives a positive test result for HIV.

A person using illegal performing enhancing drugs passes a test clearing them of drug use.

A study found a significant relationship between neck strain and jogging, when reality there is no relationship.

A healthy individual gets a positive cancer biopsy result.



# Classification metrics (an abbreviated set)

- True positive rate:

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

$$TPR = TP/P = \frac{TP}{TP+FN}$$

- AKA *sensitivity* AKA *recall*

- True negative rate:  $TNR = TN/N = \frac{TN}{FP+TN}$

- AKA *specificity*

- False positive rate:  $FPR = FP/N = \frac{FP}{FP+TN}$

- AKA *1 - specificity*

- Precision:  $PPV = \frac{TP}{TP+FP}$

- AKA *positive predictive value*

- Accuracy:  $\frac{TP+TN}{TP+TN+FP+FN}$

# Recall our model:

```
## Recall:
peng_nona %>%
  mutate(sex = if_else(sex == "female", 1, 0)) -> peng_nona_sex01
initial_fit <- glm(sex ~ ., data = peng_nona_sex01, family = "binomial")
fit <- step(initial_fit, trace = FALSE)

tibble(x = fit$linear.predictors,
        y = fit$fitted.values,
        sex = peng_nona$sex) -> fit_tibble

head(fit_tibble)
```

```
## # A tibble: 6 x 3
##       x      y sex
##   <dbl> <dbl> <fct>
## 1 -1.24  0.224 male
## 2  0.347 0.586 female
## 3  2.14  0.895 female
## 4  1.05  0.740 female
## 5 -3.89  0.0200 male
## 6  1.11  0.751 female
```

# Calculating performance measures

- Requires a *threshold* to call female/male model outcomes.
- For an example, let's say  $\geq 0.75$  is female (arbitrary "success").  $< 0.75$  is male (arbitrary "failure")
- Accuracy: 
$$\frac{TP+TN}{TP+TN+FP+FN}$$

```
threshold <- 0.75
fit_tibble %>%
  rename(truth = sex) %>%
  mutate(pred = if_else(y >= threshold, "female", "male"))
```

```
## # A tibble: 333 x 4
##       x         y truth  pred
##   <dbl>   <dbl> <fct> <chr>
## 1 -1.24  0.224   male  male
## 2  0.347  0.586   female male
## 3  2.14   0.895   female female
## 4  1.05   0.740   female male
## 5 -3.89   0.0200   male  male
## 6  1.11   0.751   female female
## 7 -8.30   0.000249 male  male
## 8  2.60   0.931   female female
## 9 -5.34   0.00477   male  male
## 10 -6.27  0.00188   male  male
## # ... with 323 more rows
```

# Detour: let's learn about `case_when()`

What if we want to do `if_else()` but with more than two options?

```
tibble(grades =  
  c(95, 88, 83, 91, 79, 72, 87))
```

```
## # A tibble: 7 x 1  
##   grades  
##   <dbl>  
## 1     95  
## 2     88  
## 3     83  
## 4     91  
## 5     79  
## 6     72  
## 7     87
```

- `>90` is "A"
- `>=80 & < 90` is "B"
- `>=70 & < 80` is "C"
- `>=60 & < 70` is "D"
- Rest is "F"

- `>90` is "A"
- `>=80 & < 90` is "B"
- `>=70 & < 80` is "C"
- `>=60 & < 70` is "D"
- Rest is "F"

```
tibble(grades = c(95, 88, 83, 91, 79, 72, 87)) %>%
  mutate(letter_grade = case_when(grades > 90 ~ "A",
                                   grades >=80 & grades < 90 ~ "B",
                                   grades >=70 & grades < 80 ~ "C",
                                   grades >=60 & grades < 70 ~ "D",
                                   TRUE ~ "F")
  ) # close mutate() parentheses
```

```
## # A tibble: 7 x 2
##   grades letter_grade
##   <dbl> <chr>
## 1     95 A
## 2     88 B
## 3     83 B
## 4     91 A
## 5     79 C
## 6     72 C
## 7     87 B
```



```

threshold <- 0.75
fit_tibble %>%
  rename(truth = sex) %>%
  mutate(pred = if_else(y >= threshold, "female", "male")) %>%
  mutate(classif = case_when(truth == "female" & pred == "female" ~ "TP",
                             truth == "female" & pred == "male" ~ "FN",
                             truth == "male" & pred == "female" ~ "TN",
                             truth == "male" & pred == "male" ~ "FP")) ->

confusion

```

```

## # A tibble: 333 x 5
##       x      y truth  pred  classif
##   <dbl> <dbl> <fct> <chr> <chr>
## 1 -1.24  0.224   male   male   FP
## 2  0.347  0.586   female male   FN
## 3  2.14   0.895   female female TP
## 4  1.05   0.740   female male   FN
## 5 -3.89   0.0200   male   male   FP
## 6  1.11   0.751   female female TP
## 7 -8.30   0.000249 male   male   FP
## 8  2.60   0.931   female female TP
## 9 -5.34   0.00477   male   male   FP
## 10 -6.27  0.00188   male   male   FP
## # ... with 323 more rows

```

```
confusion %>%  
  # how many in each classif category?  
  count(classif)
```

```
## # A tibble: 4 x 2  
##   classif      n  
##   <chr>    <int>  
## 1 FN         23  
## 2 FP        164  
## 3 TN         4  
## 4 TP       142
```

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

```
## No code, no answer:  
(142 + 4) / (142 + 4 + 164 + 23)
```

```
## [1] 0.4384384
```

```
confusion %>%  
  count(classif) %>%  
  pivot_wider(names_from = classif, values_from = n)
```

```
## # A tibble: 1 x 4  
##       FN    FP    TN    TP  
##   <int> <int> <int> <int>  
## 1     23   164     4   142
```

```
confusion %>%  
  count(classif) %>%  
  pivot_wider(names_from = "classif", values_from = "n") %>%  
  mutate(accuracy = (TP + TN)/(TP + TN + FP + FN)) %>%  
  pull(accuracy)
```

```
## [1] 0.4384384
```

# How good is the model?

- In linear regression, we can gauge the model performance with  $R^2$  and RMSE.
- In logistic regression, performance **depends** on your chosen threshold!  
So, how do we choose a threshold?
  - Usually, find the threshold that makes the false positive rate <5%>
- We also use **AUC** (area under the curve... what curve?)

# Evaluating logistic regressions

## Receiver Operating Characteristic Curve

- TPR on Y-axis
- FPR (1 - specificity) on X-axis
- The AUC (area under the curve) is an overall assessment of performance *at any threshold*

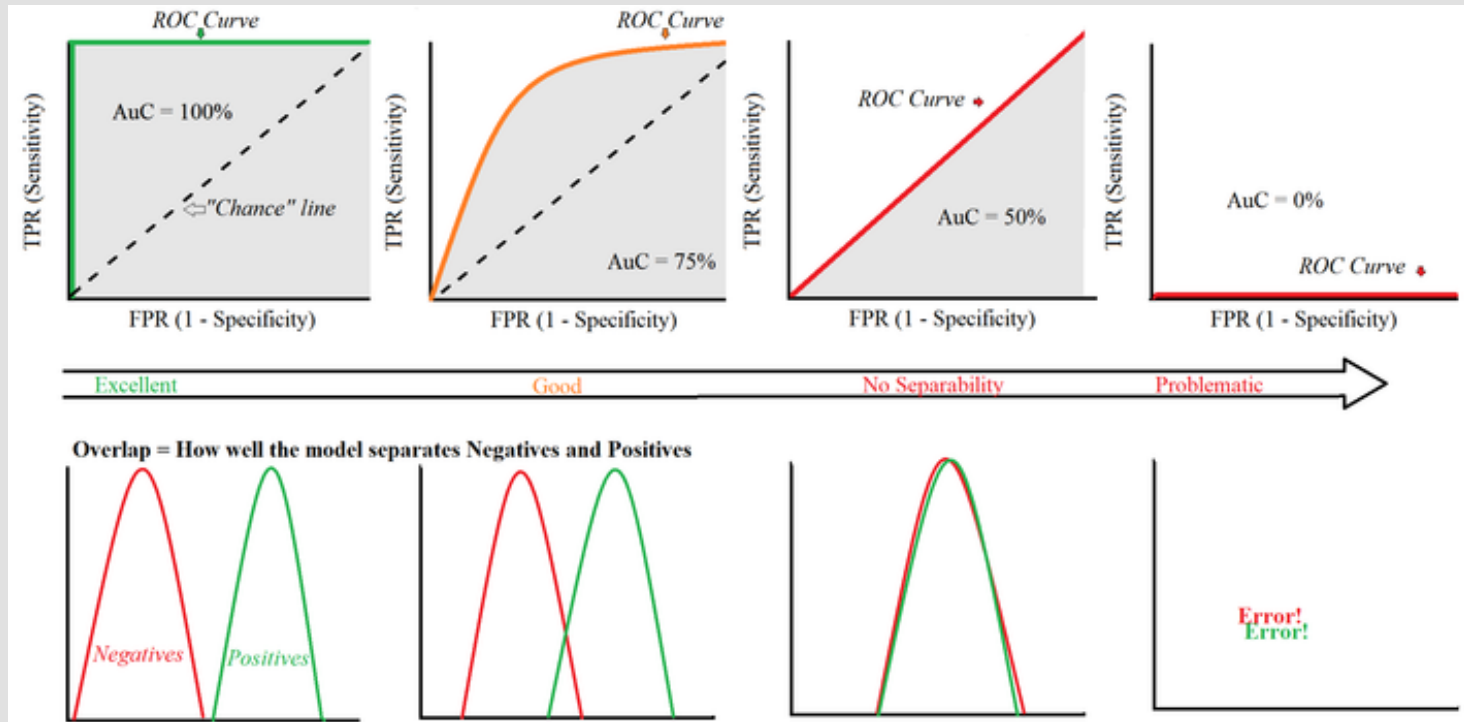
- $TPR = TP/P = \frac{TP}{TP+FN}$   
(*sensitivity AKA recall*)

- $TNR = TN/N = \frac{TN}{FP+TN}$   
(*specificity*)

- $FPR = FP/N = \frac{FP}{FP+TN}$  (1 - specificity)

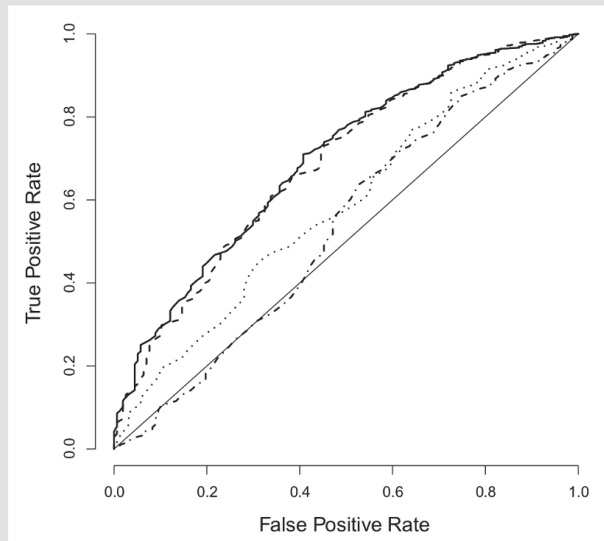


# Getting a "feel" for ROC curves

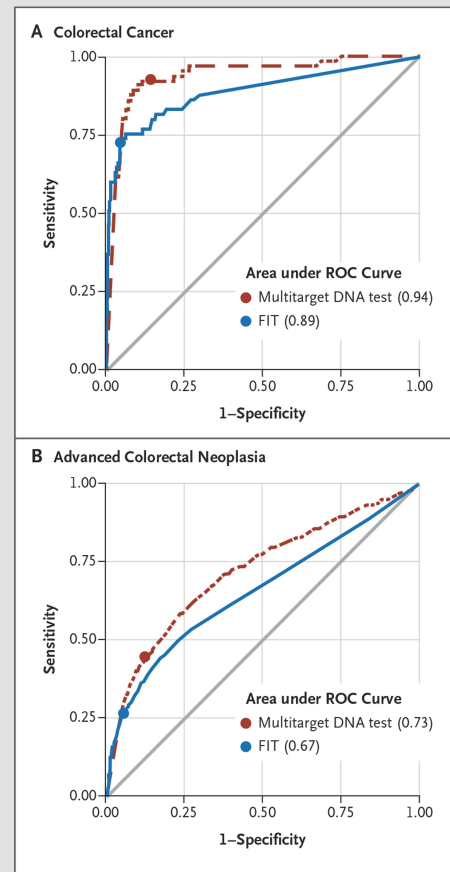


# Examples of ROC curves in the literature

Keller et al. Genome Biol Evol 2012;  
4:80-88



Imperiale et al. N Engl J Med 2014;  
370:1287-1297



# ROC vs PR

- ROC curves are suitable when data is *balanced*
  - Similar amounts of positives, negatives in the dataset
  - FPR (1 - specificity) on X-axis, TPR on Y-axis
- **Precision-Recall** curves are more suitable for *unbalanced* data
  - Precision (PPV) on Y-axis, recall (TPR) on X-axis

- $TPR = TP/P = \frac{TP}{TP+FN}$  (*recall*)
- $FPR = FP/N = \frac{FP}{FP+TN}$
- $PPV = \frac{TP}{TP+FP}$

# Is the penguin data balanced? Yes.

```
peng_nona %>%  
  count(sex)
```

```
## # A tibble: 2 x 2  
##   sex      n  
##   <fct> <int>  
## 1 female  165  
## 2 male   168
```

- *Problematically imbalanced* would be 4000 females and 5 males, or vice versa.

# Making ROC curves

- Recall:
  - Our model fit is saved in `fit`
  - Our model was built with `peng_nona_sex01` dataset

```
## Use the pROC library to help you - installed already in cloud!  
library(pROC)
```

```
## Use the function roc()!!  
model_roc <- roc(peng_nona_sex01$sex, fit$linear.predictors)
```

```
## This also works the same:  
model_roc <- roc(peng_nona_sex01$sex, fit$fitted.values)
```

# Getting information out

```
model_roc$auc
```

```
## Area under the curve: 0.977
```

**Models are usually not this good.**

```
#Piped into head() to fit on the slide
```

```
## True positive rates: The x-axis!!  
model_roc$sensitivities %>% head()
```

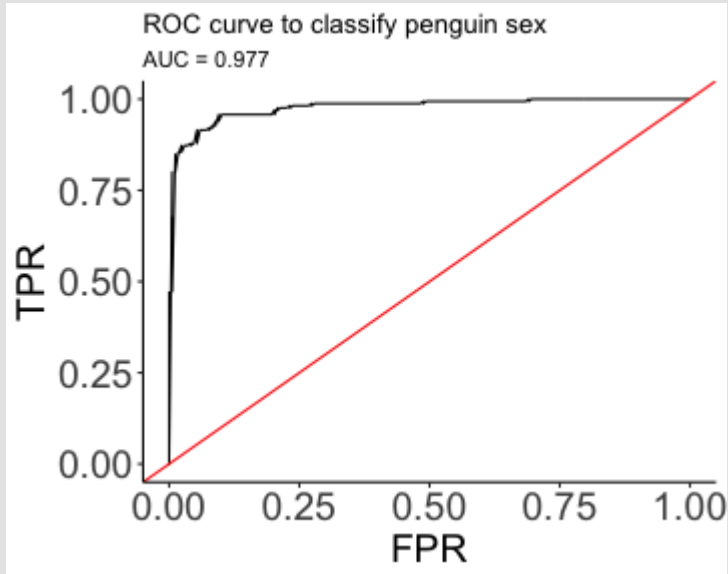
```
## [1] 1 1 1 1 1 1
```

```
## False positives rates: The y-axis!!  
1 - model_roc$specificities %>% head()
```

```
## [1] 1.0000000 0.9940476 0.9880952 0.9821429 0.9761905  
## [6] 0.9702381
```

# Make an ROC curve

```
tibble(TPR = model_roc$sensitivities,  
       FPR = 1 - model_roc$specificities) %>%  
  ggplot(aes(x = FPR, y = TPR)) +  
  geom_line() +  
  labs(title = "ROC curve to classify penguin sex",  
       subtitle = paste("AUC =", round(model_roc$auc, 3)) ) +  
  ## this is the y=x line to GUIDE US and help us interpret the curve  
  geom_abline(col = "red")
```



# Making predictions

This is the same as running the model on new data it has never seen (this is important!)

```
## To remind you: What were the predictors?  
tidy(fit)
```

```
## # A tibble: 6 x 5  
##   term                estimate std.error statistic  p.value  
##   <chr>              <dbl>    <dbl>    <dbl>    <dbl>  
## 1 (Intercept)       76.1      9.80      7.76 8.34e-15  
## 2 speciesChinstr...  6.90      1.56      4.43 9.56e- 6  
## 3 speciesGentoo     7.88      2.25      3.50 4.62e- 4  
## 4 bill_len          -0.620     0.131     -4.74 2.11e- 6  
## 5 bill_dep          -1.64      0.331     -4.97 6.69e- 7  
## 6 mass              -0.00596   0.00106    -5.62 1.88e- 8
```



# Making predictions

```
what_are_you_new_penguin <- tibble(  
  species = "Gentoo",  
  bill_len = 41.5,  
  bill_dep = 13.6,  
  mass     = 3850  
)  
  
## NEED `type = "response"` to get a probability out  
predict(fit, what_are_you_new_penguin, type = "response")
```

```
##           1  
## 0.9999976
```

# Validating with training/testing

```
set.seed(1) # reproducibility!!

training_frac <- 0.6

## Training and testing data splits
peng_nona_sex01 %>%
  sample_frac(training_frac) -> train_data
anti_join(peng_nona_sex01, train_data) -> test_data

## Fit to training
our_formula <- fit$call$formula # previously made!
train_fit <- glm(our_formula, data = train_data, family = "binomial")

## How's the training data model?
training_roc <- roc(train_data$sex, train_fit$fitted.values)
training_roc$auc
```

```
## Area under the curve: 0.9874
```

We have to take an extra step for logistic (vs linear) regression and fit the model to test data ourselves, since we don't have any nice `modelr` helper functions like `rsquare()` and `rmse()`.

```
## Fit the model to the testing data  
test_fit <- predict(train_fit, test_data, type = "response")  
  
## How's the testing data model?  
testing_roc <- roc(test_data$sex, test_fit)  
testing_roc$auc
```

`## Area under the curve: 0.9588`

All together now:

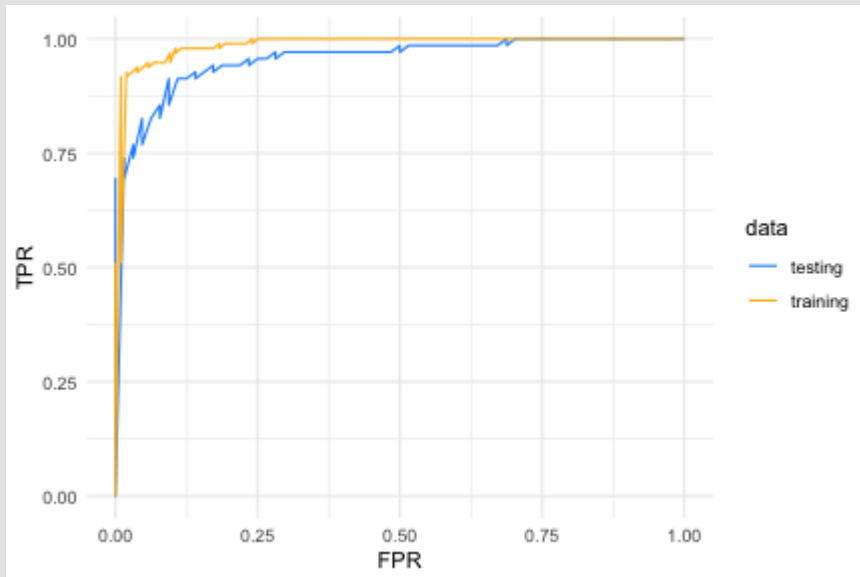
Data	AUC
Training	0.9874
Testing	0.9588

*Seem this model performs pretty darn well.*

# One ROC curve to rule them all.

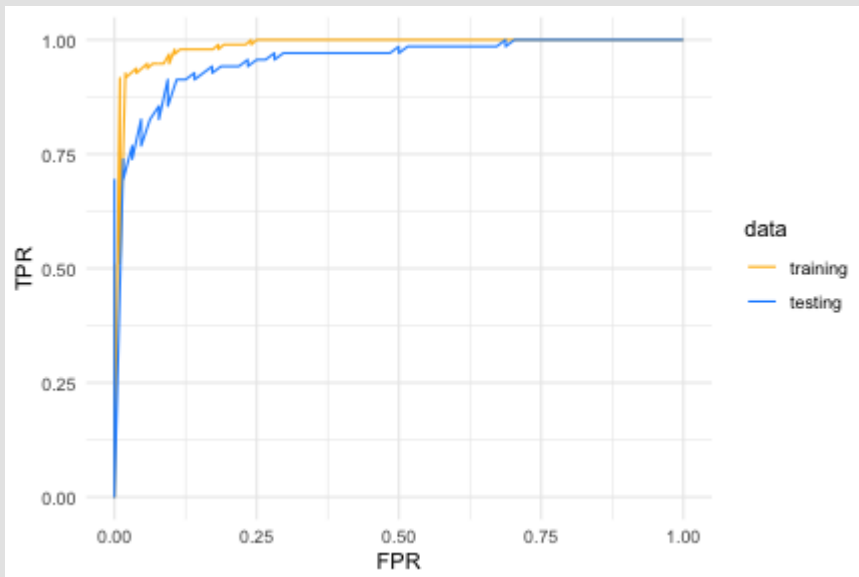
```
train_and_test <- tibble(  
  TPR = c(training_roc$sensitivities, testing_roc$sensitivities),  
  FPR = c(1 - training_roc$specificities, 1 - testing_roc$specificities),  
  data = c(rep("training", length(training_roc$sensitivities)),  
            rep("testing", length(testing_roc$sensitivities))  
)
```

```
ggplot(train_and_test) +  
  aes(x = FPR, y = TPR, color = data) +  
  geom_line() +  
  #fun!  
  scale_color_manual(values = c("dodgerblue", "darkgoldenrod1")) +  
  theme_minimal()
```



# Don't violate best practices!

```
ggplot(train_and_test) +  
  aes(x = FPR, y = TPR,  
      color = fct_relevel(data, c("training", "testing")))) +  
  geom_line() +  
  scale_color_manual(name = "data",  
                    values = c("darkgoldenrod1", "dodgerblue")) +  
  theme_minimal()
```



# Final words of wisdom

- Visualizing the model = the literal logistic curve
  - Visualizing the model performance = ROC curve
- 
- In the ROC curves you may have noticed a weird spike at the beginning. Don't worry about it. It's a consequence of using the pROC package.