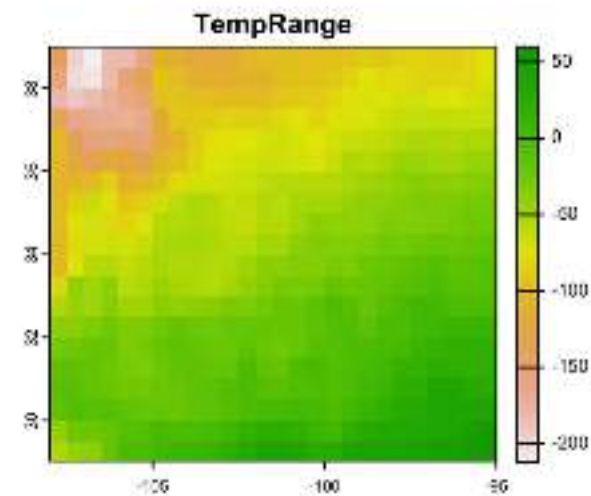
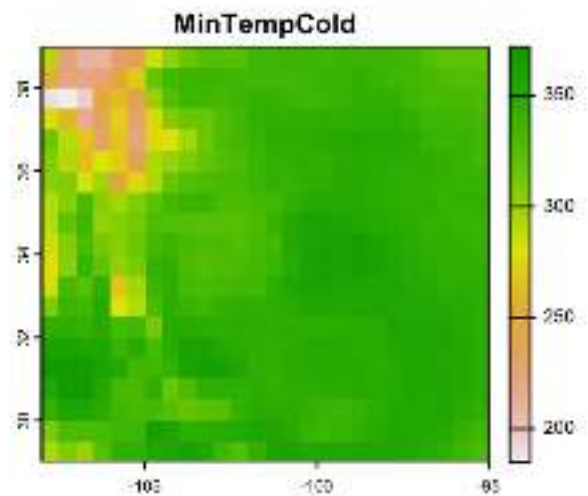
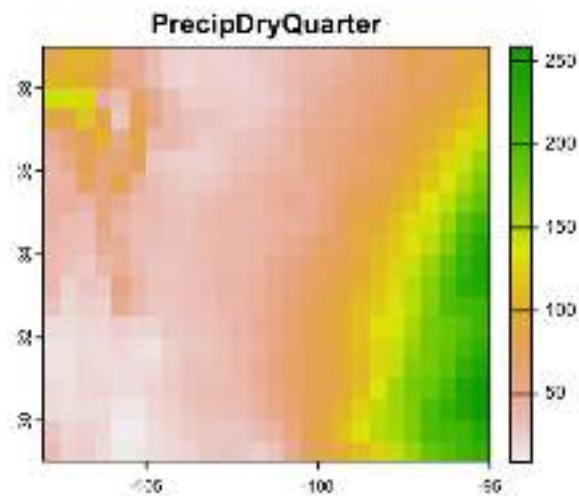
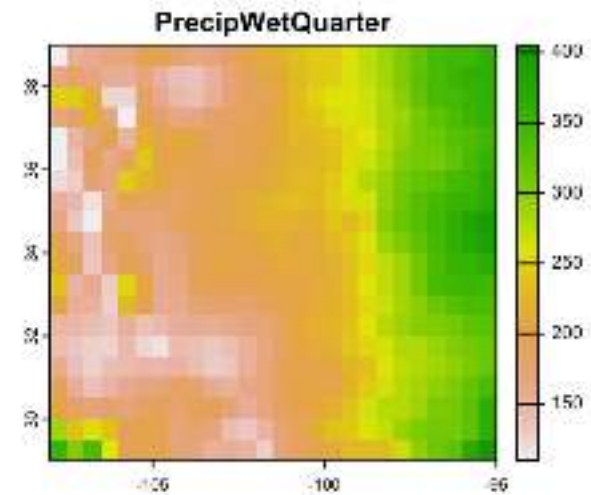
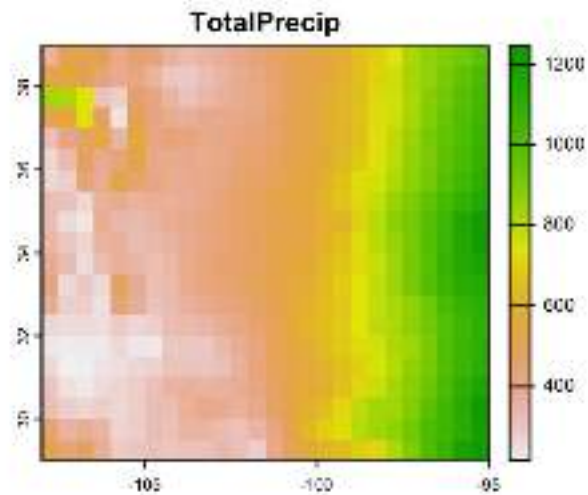
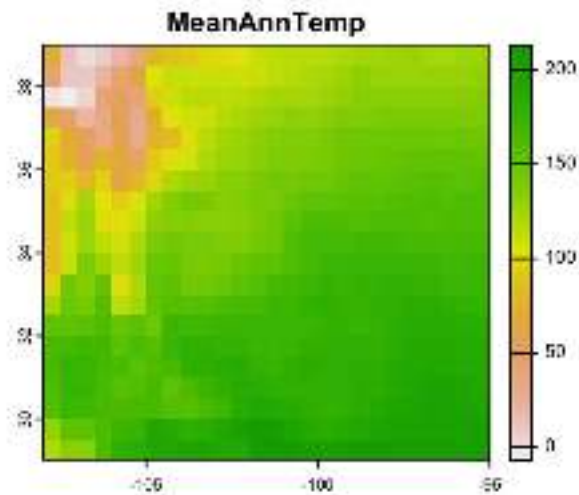


Movement and Networks I

HES 505 Fall 2023: Session 25

Matt Williamson

Revisiting Deviance



Pseudo- R^2

$$R_L^2 = \frac{D_{\text{null}} - D_{\text{fitted}}}{D_{\text{null}}}$$

Cohen's Likelihood Ratio

```
1 logistic.rich <- glm(y ~ MeanAnnTemp + PrecipWetQuarter + PrecipDryQuarter,  
2                     family=binomial(link="logit"),  
3                     data=pts.df[,2:8])  
4  
5 logistic.null <- glm(y ~ 1,  
6                     family=binomial(link="logit"),  
7                     data=pts.df[,2:8])  
8 with(logistic.rich,  
9       null.deviance - deviance)/with(logistic.rich,  
10                                       null.deviance)
```

```
[1] 0.4495966
```

```
1 1 - exp(2*(logLik(logistic.null)[1] - logLik(logistic.rich)[1])/nobs(logist
```

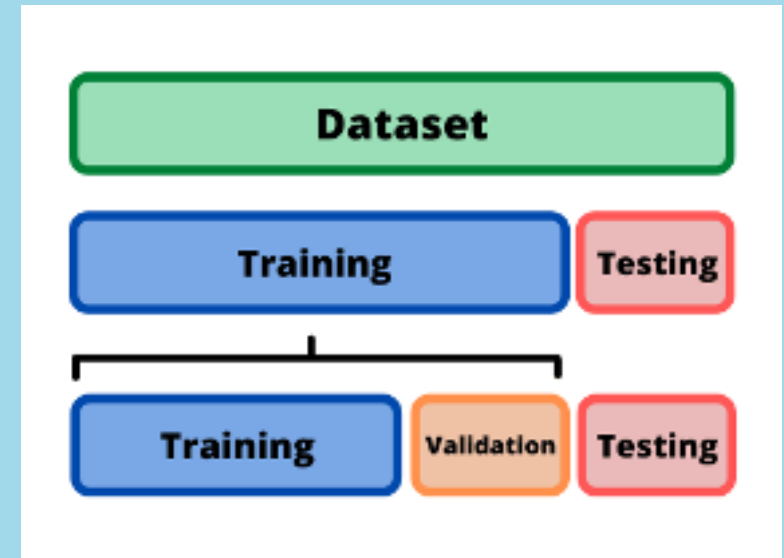
```
[1] 0.4308873
```

So what is deviance???

- Saturated model (i.e., perfect model)
- One parameter for each observation
- Null model
- Only 1 parameter (for the intercept)
- Your model
- 1 parameter for each covariate

Sub-sampling Methods

- Split data into *training* and *testing*
- Testing set needs to be large enough for results to be statistically meaningful
- Test set should be representative of the data as a whole
- Validation data used to tune parameters (not always)



Subsampling your data with **caret**

```
1 pts.df$y <- as.factor(ifelse(pts.df$y == 1, "Yes", "No"))
2 library(caret)
3 Train <- createDataPartition(pts.df$y, p=0.6, list=FALSE)
4
5 training <- pts.df[ Train, ]
6 testing <- pts.df[ -Train, ]
```


Misclassification

- Confusion matrices compare actual values to predictions
- True Positive (TN) - This is correctly classified as the class of interest / target.
- True Negative (TN) - This is correctly classified as not a class of interest / target.
- False Positive (FP) - This is wrongly classified as the class of interest / target.
- False Negative (FN) - This is wrongly classified as not a class of interest / target.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Confusion Matrices in R

```
1 train.log <- glm(y ~ .,  
2                 family="binomial"  
3                 data=training[,2:  
4  
5 predicted.log <- predict(train.log  
6                         newdata=t  
7                         type="res  
8  
9 pred <- as.factor(  
10   ifelse(predicted.log > 0.5,  
11           "Yes",  
12           "No" ))
```

```
1 confusionMatrix(testing$y, pred)
```

Confusion Matrix and Statistics

	Reference	
Prediction	No	Yes
No	27	0
Yes	5	7

Accuracy : 0.8718
95% CI : (0.7257, 0.957)
No Information Rate : 0.8205
P-Value [Acc > NIR] : 0.27535

Kappa : 0.6597

Mcnemar's Test P-Value : 0.07364

Sensitivity : 0.8438
Specificity : 1.0000
Pos Pred Value : 1.0000
Neg Pred Value : 0.5833
Prevalence : 0.8205
Detection Rate : 0.6923

Confusion Matrices

**Depends upon
threshold!!**

Confusion Matrices in R

```
1 library(tree)
2 tree.model <- tree(y ~ . , training)
3 predict.tree <- predict(tree.model,
```

```
1 confusionMatrix(testing$y, predict.tree)
```

Confusion Matrix and Statistics

	Reference	
Prediction	No	Yes
No	27	0
Yes	5	7

Accuracy : 0.8718
95% CI : (0.7257, 0.957)
No Information Rate : 0.8205
P-Value [Acc > NIR] : 0.27535

Kappa : 0.6597

Mcnemar's Test P-Value : 0.07364

Sensitivity : 0.8438
Specificity : 1.0000
Pos Pred Value : 1.0000
Neg Pred Value : 0.5833
Prevalence : 0.8205
Detection Rate : 0.6923

Confusion Matrices in R

```
1 library(randomForest)
2 class.model <- y ~ .
3 rf <- randomForest(class.model, data)
4 predict.rf <- predict(rf, newdata=)
```

```
1 confusionMatrix(testing$y, predict.rf)
```

Confusion Matrix and Statistics

	Reference	
Prediction	No	Yes
No	27	0
Yes	8	4

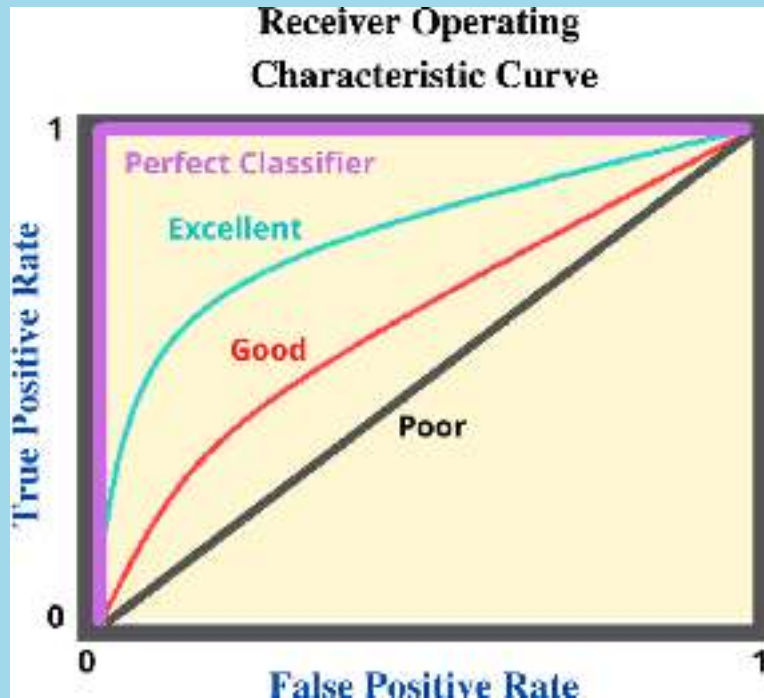
Accuracy : 0.7949
95% CI : (0.6354, 0.907)
No Information Rate : 0.8974
P-Value [Acc > NIR] : 0.98465

Kappa : 0.4091

Mcnemar's Test P-Value : 0.01333

Sensitivity : 0.7714
Specificity : 1.0000
Pos Pred Value : 1.0000
Neg Pred Value : 0.3333
Prevalence : 0.8974
Detection Rate : 0.6923

Threshold-Free Methods



- Receiver Operating Characteristic Curves
- Illustrates discrimination of binary classifier as the threshold is varied
- Area Under the Curve (AUC) provides an estimate of classification ability

Criticisms of ROC/AUC

- Treats false positives and false negatives equally
- Undervalues models that predict across smaller geographies
- Focus on *discrimination* and not *calibration*
- New methods for presence-only data

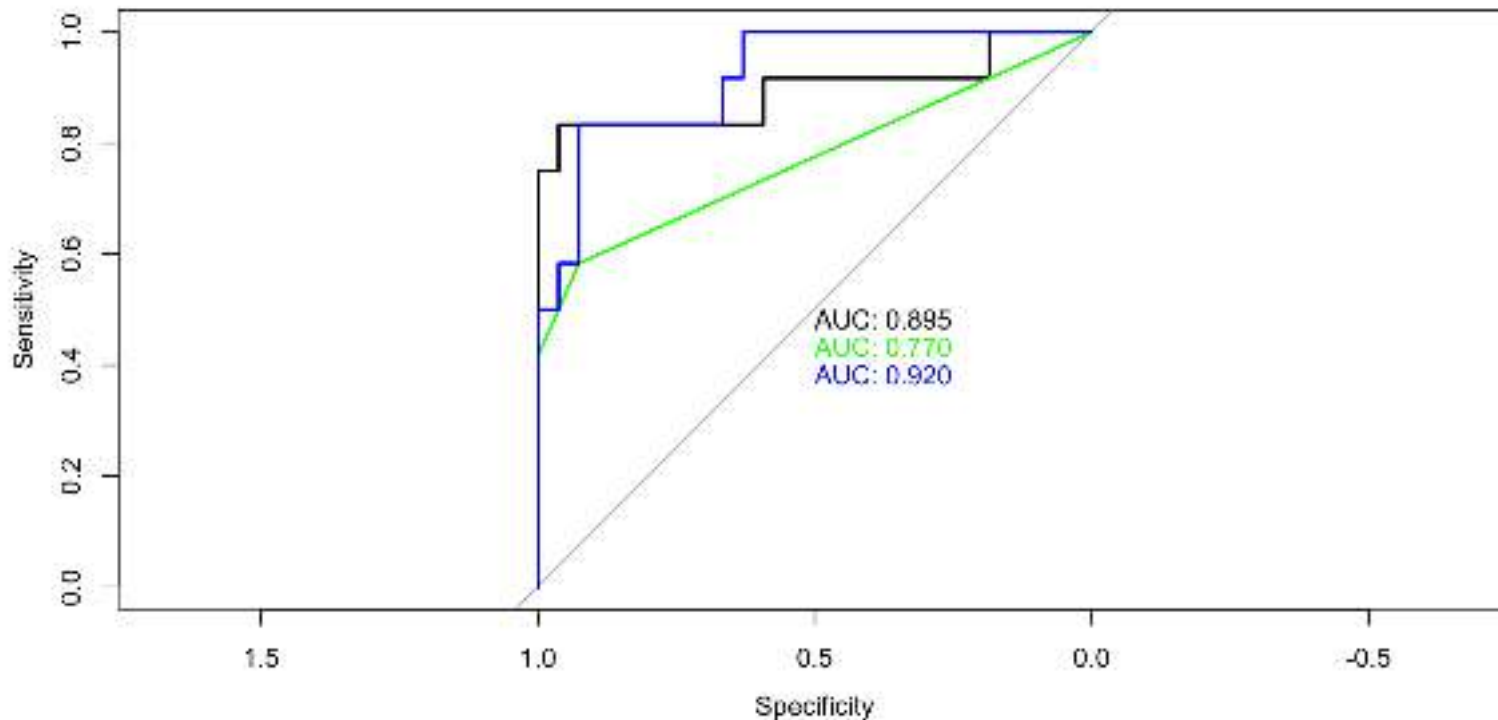
ROC in R (using **pROC**)

- Generate predictions (note the difference for tree and rf)

```
1 library(pROC)
2 train.log <- glm(y ~ .,
3                 family="binomial",
4                 data=training[,2:8])
5
6 predicted.log <- predict(train.log,
7                          newdata=testing[,2:8],
8                          type="response")
9
10 predict.tree <- predict(tree.model, newdata=testing[,2:8], type="vector")[,
11
12 predict.rf <- predict(rf, newdata=testing[,2:8], type="prob")[,2]
```


ROC in R (using pROC)

```
1 plot(roc(testing$y, predicted.log), print.auc=TRUE)
2
3 plot(roc(testing$y, predict.tree), print.auc=TRUE, print.auc.y = 0.45, col=
4
5 plot(roc(testing$y, predict.rf), print.auc=TRUE, print.auc.y = 0.4, col="bl
```



Cross-validation

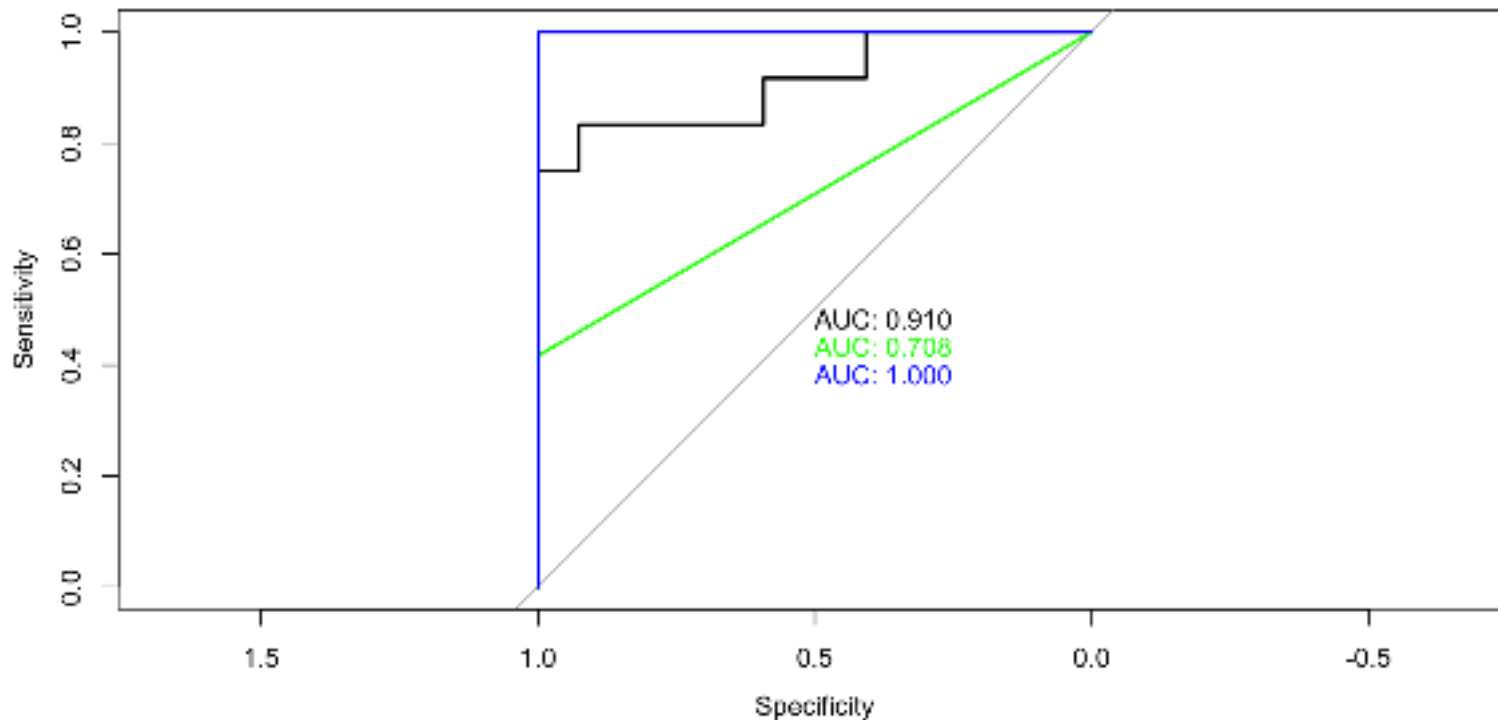
- Often want to make sure that fit/accuracy not a function of partition choice
- Cross-validation allows resampling of data (multiple times)
- K-fold - Data are split into K datasets of \sim equal size, model fit to observations to predict heldout set
- Leave One Out (LOO) - Model fit to $n-1$ observations to predict the held out observation

Crossvalidation in R using caret

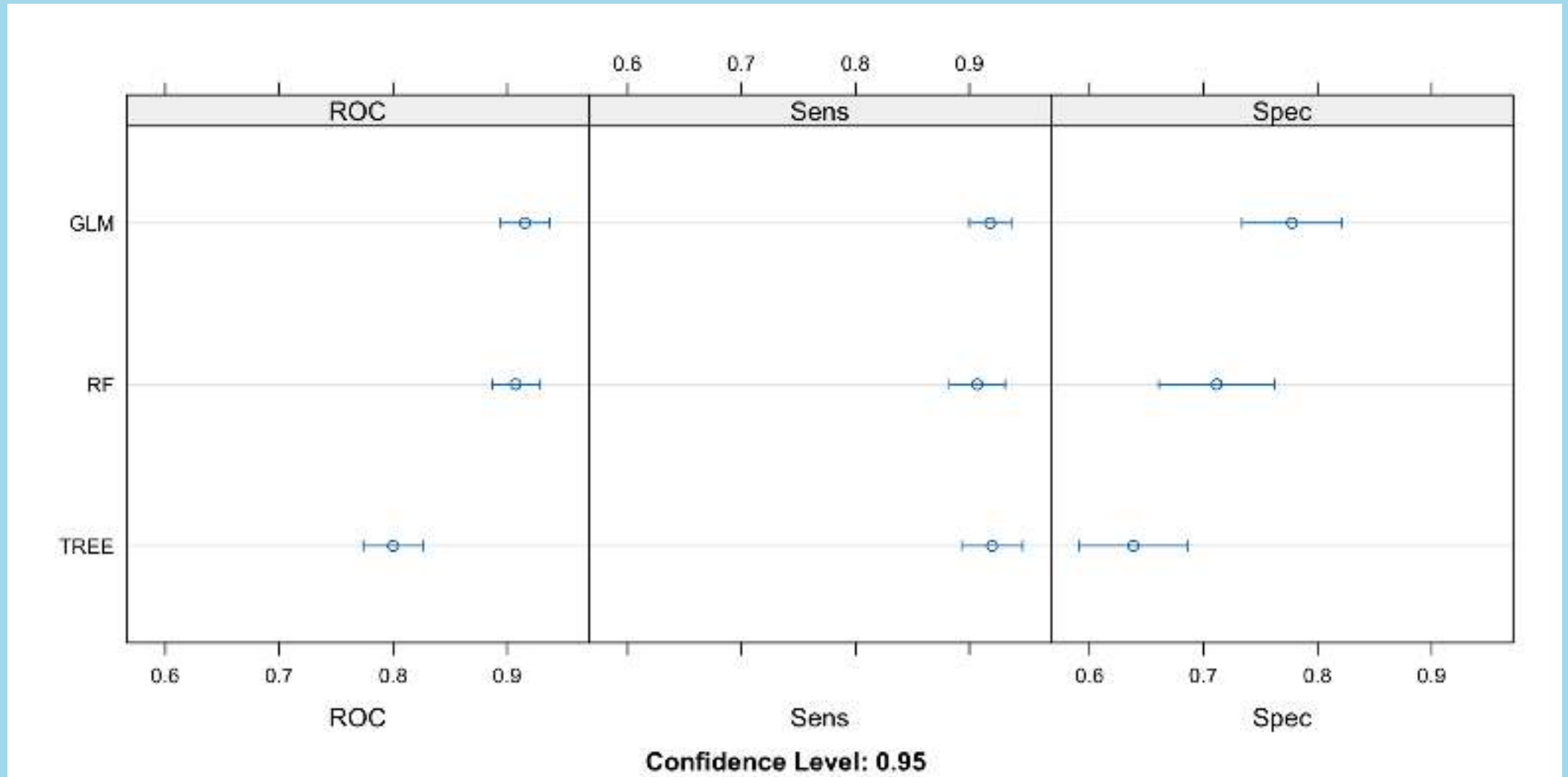
```
1 fitControl <- trainControl(method = "repeatedcv",
2                             number = 10,
3                             repeats = 10,
4                             classProbs = TRUE,
5                             summaryFunction = twoClassSummary)
6
7 log.model <- train(y ~., data = pts.df[,2:8],
8                   method = "glm",
9                   trControl = fitControl)
10 pred.log <- predict(log.model, newdata = testing[,2:8], type="prob")[,2]
11
12 tree.model <- train(y ~., data = pts.df[,2:8],
13                   method = "rpart",
14                   trControl = fitControl)
15
16 pred.tree <- predict(tree.model, newdata=testing[,2:8], type="prob")[,2]
17
18 rf.model <- train(v ~., data = pts.df[,2:8],
```

Crossvalidation in **R** using **caret**

```
1 plot(roc(testing$y, pred.log), print.auc=TRUE)
2
3 plot(roc(testing$y, pred.tree), print.auc=TRUE, print.auc.y = 0.45, col="gr
4
5 plot(roc(testing$y, pred.rf), print.auc=TRUE, print.auc.y = 0.4, col="blue"
```



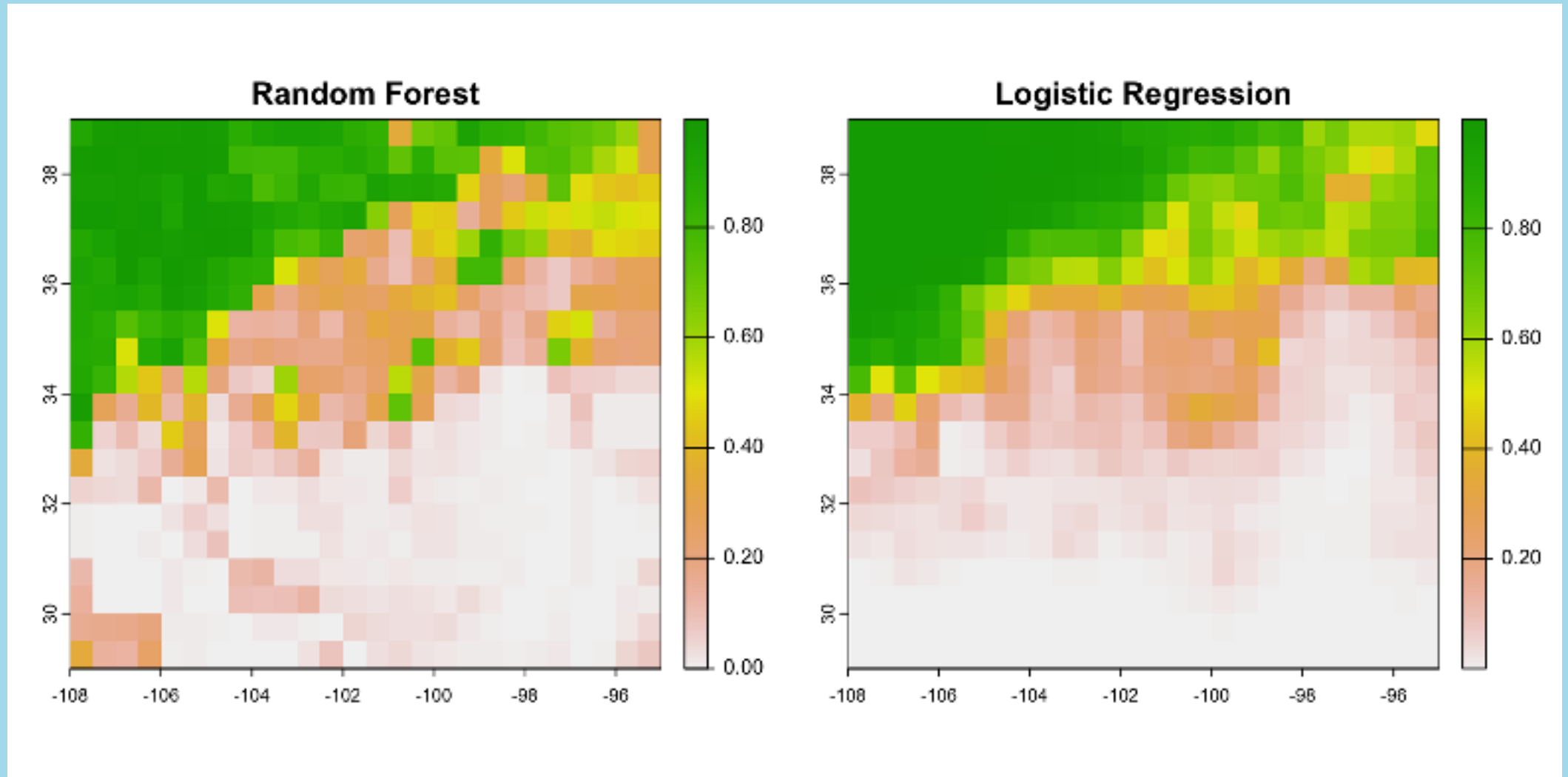
Crossvalidation in **R** using **caret**



Spatial predictions

```
1 best.rf <- rf.model$finalModel
2 best.glm <- log.model$finalModel
3
4 rf.spatial <- terra::predict(pred.stack.scl, best.rf, type="prob")
5
6
7 glm.spatial <- terra::predict(pred.stack.scl, best.glm, type="response" )
```

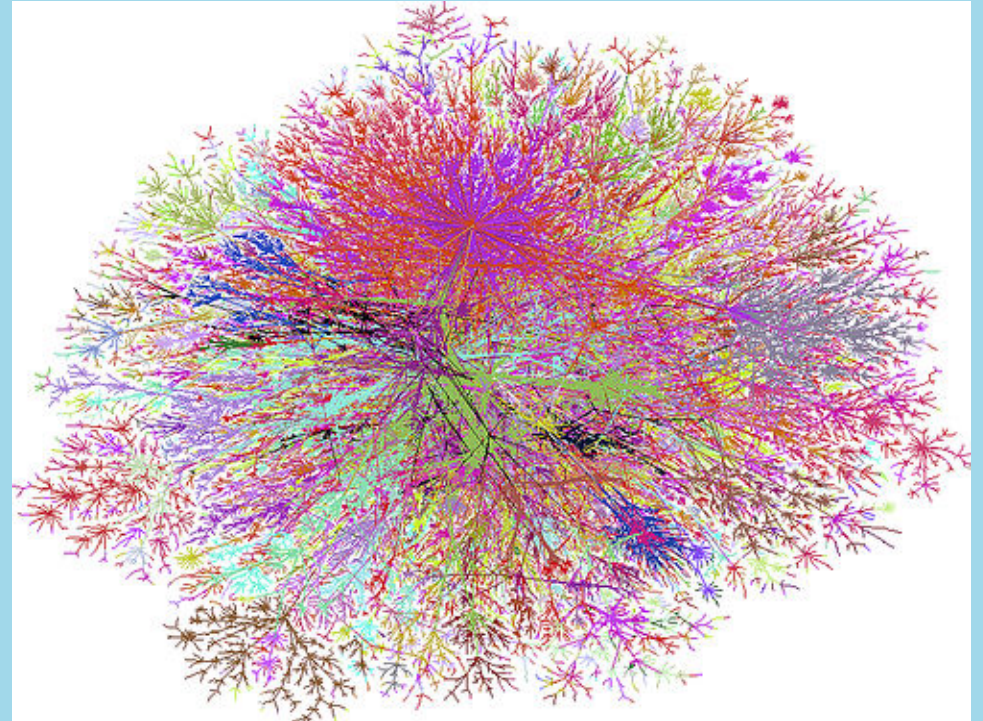
Spatial predictions



On to networks!

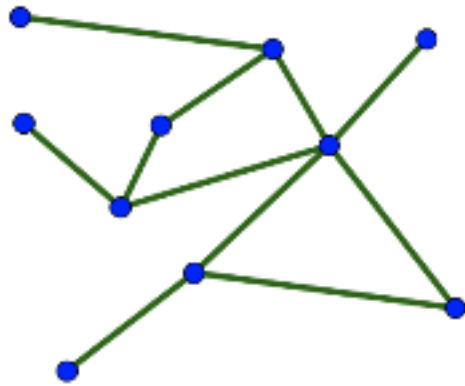
What is a network?

- A collection of connected objects
- Tend to be described in terms of nodes (the objects) and edges (the connections)
- Analyzed using algorithms from graph theory

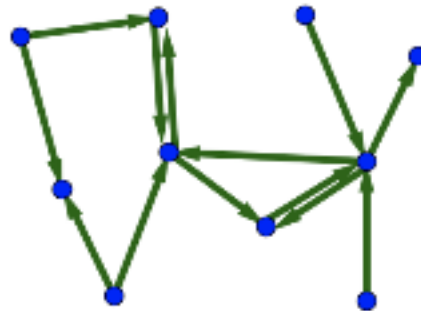


Types of networks

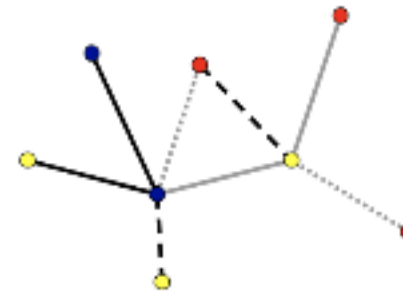
- (Un)directed
- Weighted
- Multi-type



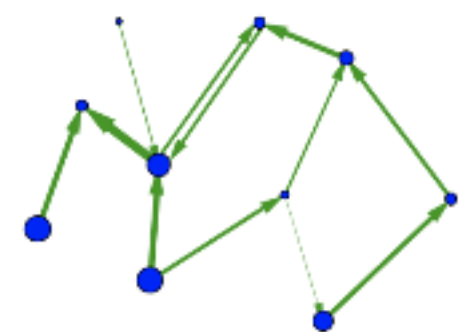
An undirected network.



A directed network.

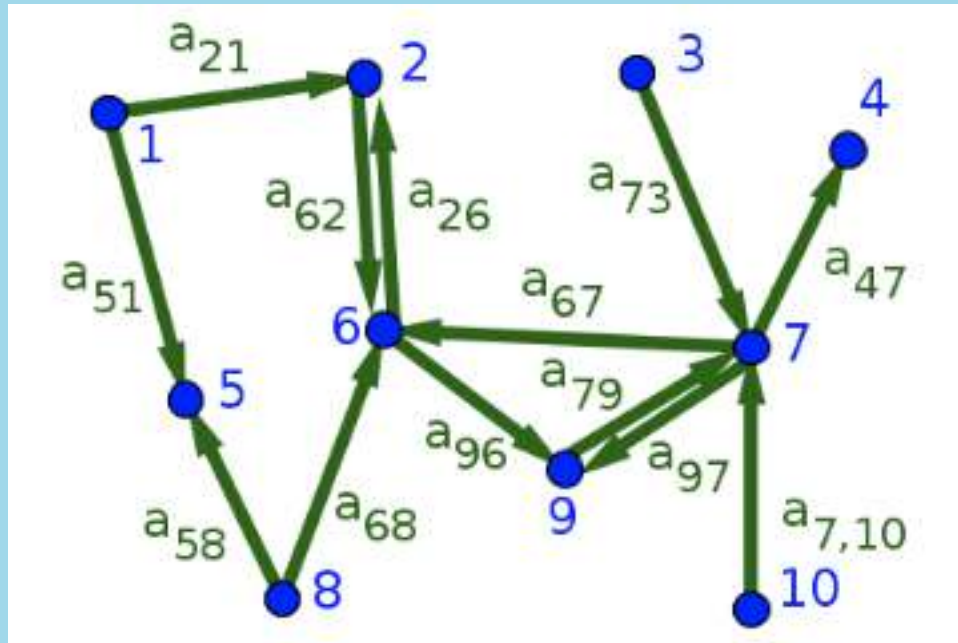


An undirected network where the nodes and edges have different types, as indicated by their colors and line styles.



A directed network where the edges and nodes have different weights, as indicated by their sizes.

Describing networks for analysis



$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Common measures

- Graph-level: density, diameter, distance
- Component-level: density, distribution
- Node-level: centrality, degree-distribution

Common questions

- What are the shortest paths across the network?
- Where are the most important locations for maintaining the network?
- How does the loss of a node alter the subsequent configuration of the network?
- How do we translate typical movement paths into network structures?

