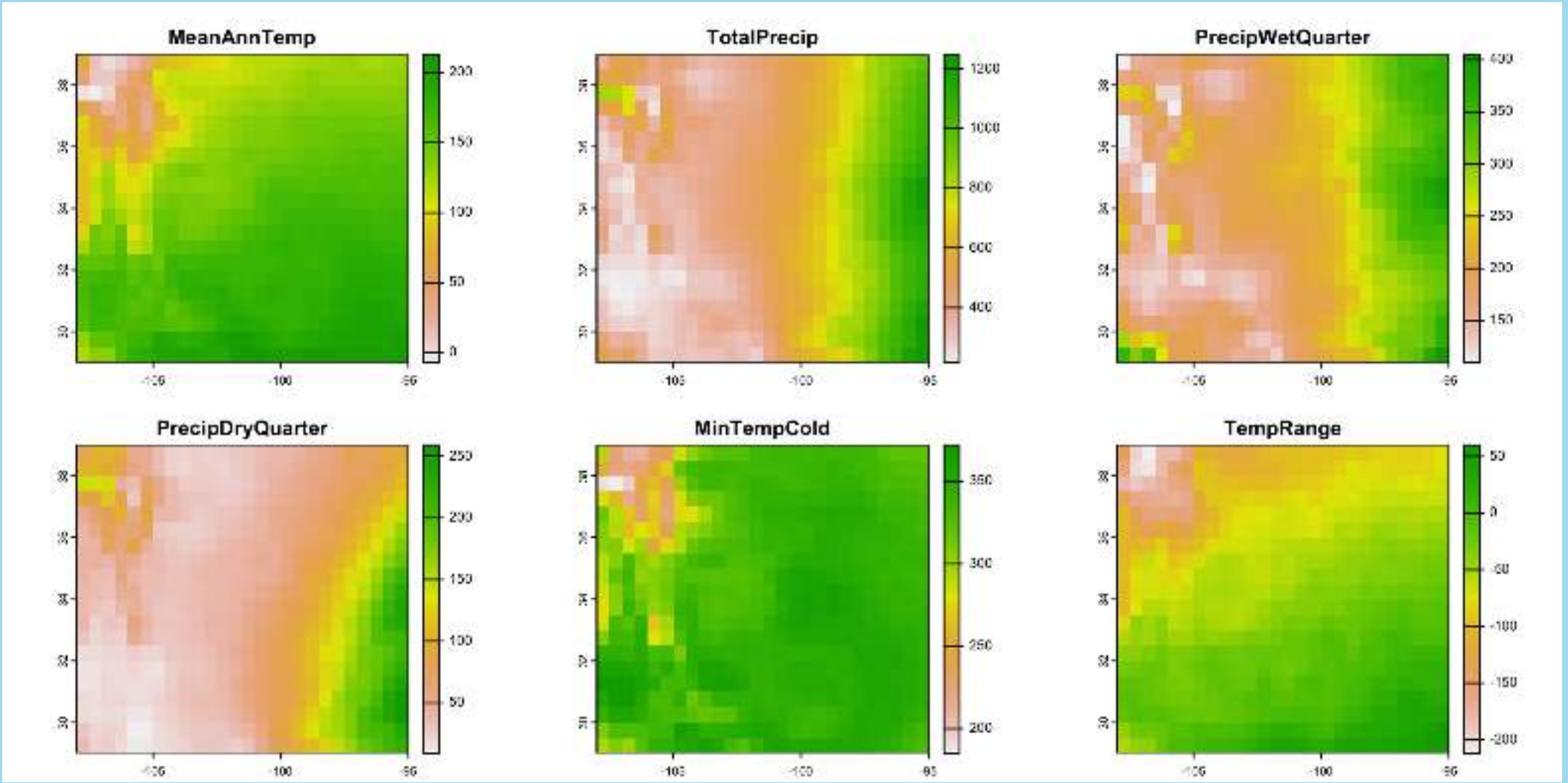
Movement and Networks I

HES 505 Fall 2023: Session 25

Matt Williamson

Revisiting Deviance


MeanAnnTemp
TotalPrecip
PrecipWetQuarter
PrecipDryQuarter
MinTempCold
TempRange

Pseudo- R^2

$$R^2_L = \frac{D_{null} - D_{fitted}}{D_{null}}$$

$$R^2_{CS} = 1 - \left(\frac{L_0}{L_M}\right)^{(2/n)}$$

$$= 1 - \exp^{2(\ln(L_0) - \ln(L_M))/n}$$

Cohen's Likelihood Ratio

```
logistic.rich <- glm(y ~ MeanAnnTemp + PrecipWetQuarter + PrecipDryQuarter,
                     family=binomial(link="logit"),
                     data=pts.df[,2:8])

logistic.null <- glm(y ~ 1,
                     family=binomial(link="logit"),
                     data=pts.df[,2:8])
with(logistic.rich,
     null.deviance - deviance)/with(logistic.rich,
                                    null.deviance)
```

```
[1] 0.4495966
```

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

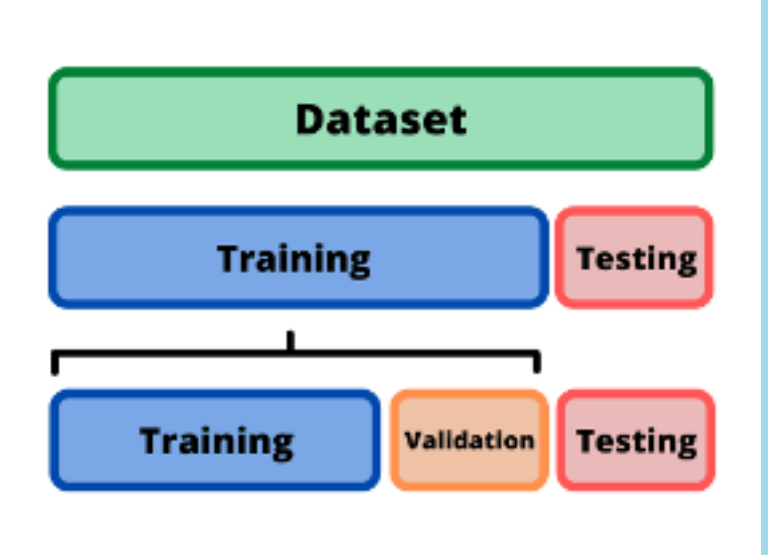
```
[1] 0.4308873
```

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)

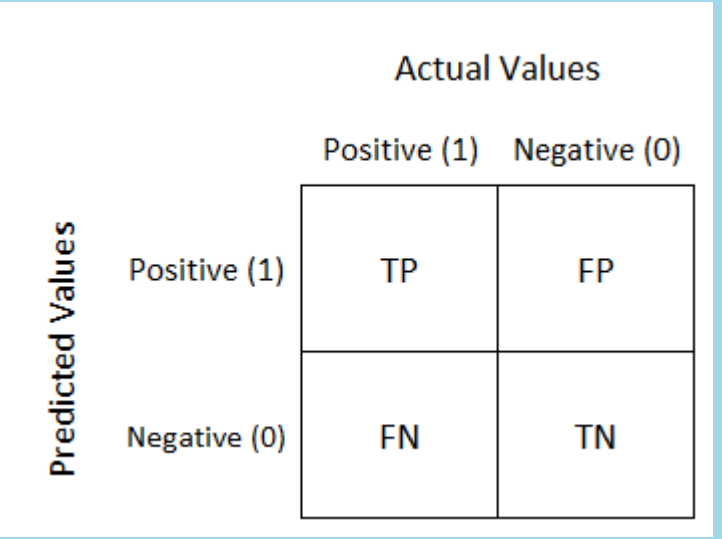


Subsampling your data with caret

```
pts.df$y <- as.factor(ifelse(pts.df$y == 1, "Yes", "No"))
library(caret)
Train <- createDataPartition(pts.df$y, p=0.6, list=FALSE)

training <- pts.df[ Train, ]
testing <- pts.df[ -Train, ]
```

Misclassification

- Confusion matrices compare actual values to predictions



- True Positive (TN) - This is correctly classified as the class if 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.

Confusion Matrices in R

```
train.log <- glm(y ~ .,
                 family="binomial"
                 data=training[,2:

predicted.log <- predict(train.log
                         newdata=t
                         type="res

pred <- as.factor(
  ifelse(predicted.log > 0.5,
                         "Yes",
                         "No"))
```

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

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  23   4
       Yes  5   7

               Accuracy : 0.7692
                 95% CI : (0.6067, 0.8887)
    No Information Rate : 0.7179
    P-Value [Acc > NIR] : 0.3037

                  Kappa : 0.4455

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.8214
            Specificity : 0.6364
         Pos Pred Value : 0.8519
         Neg Pred Value : 0.5833
             Prevalence : 0.7179
         Detection Rate : 0.5897
```

Confusion Matrices

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

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{FP + TN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Depends upon threshold!!

Confusion Matrices in R

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

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

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  22   5
       Yes  5   7

               Accuracy : 0.7436
                 95% CI : (0.5787, 0.8696)
    No Information Rate : 0.6923
    P-Value [Acc > NIR] : 0.3075

                  Kappa : 0.3981

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.8148
            Specificity : 0.5833
         Pos Pred Value : 0.8148
         Neg Pred Value : 0.5833
             Prevalence : 0.6923
         Detection Rate : 0.5641
```

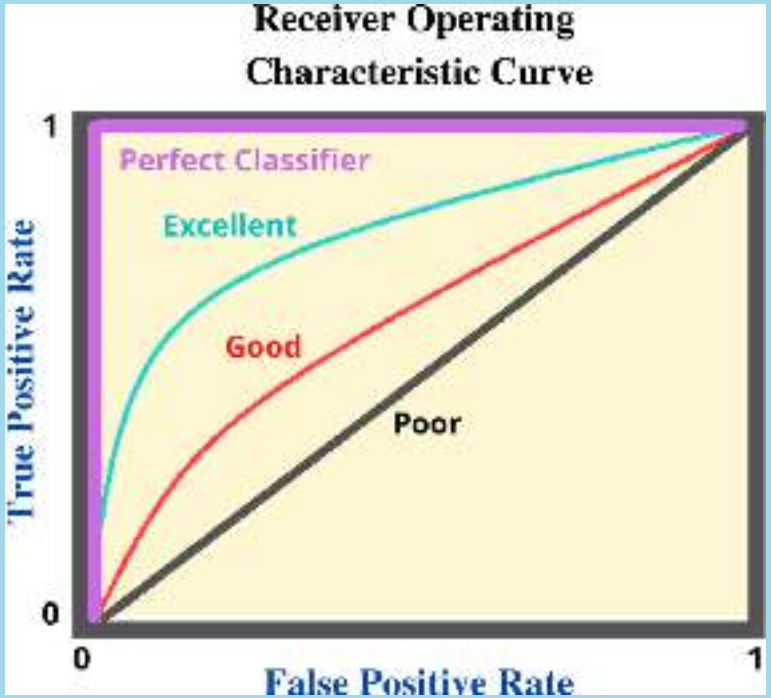
Confusion Matrices in R

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

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

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  20   7
       Yes  4   8

               Accuracy : 0.7179
                 95% CI : (0.5513, 0.85)
    No Information Rate : 0.6154
    P-Value [Acc > NIR] : 0.1236

                  Kappa : 0.381

 Mcnemar's Test P-Value : 0.5465

            Sensitivity : 0.8333
            Specificity : 0.5333
         Pos Pred Value : 0.7407
         Neg Pred Value : 0.6667
             Prevalence : 0.6154
         Detection Rate : 0.5128
```

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

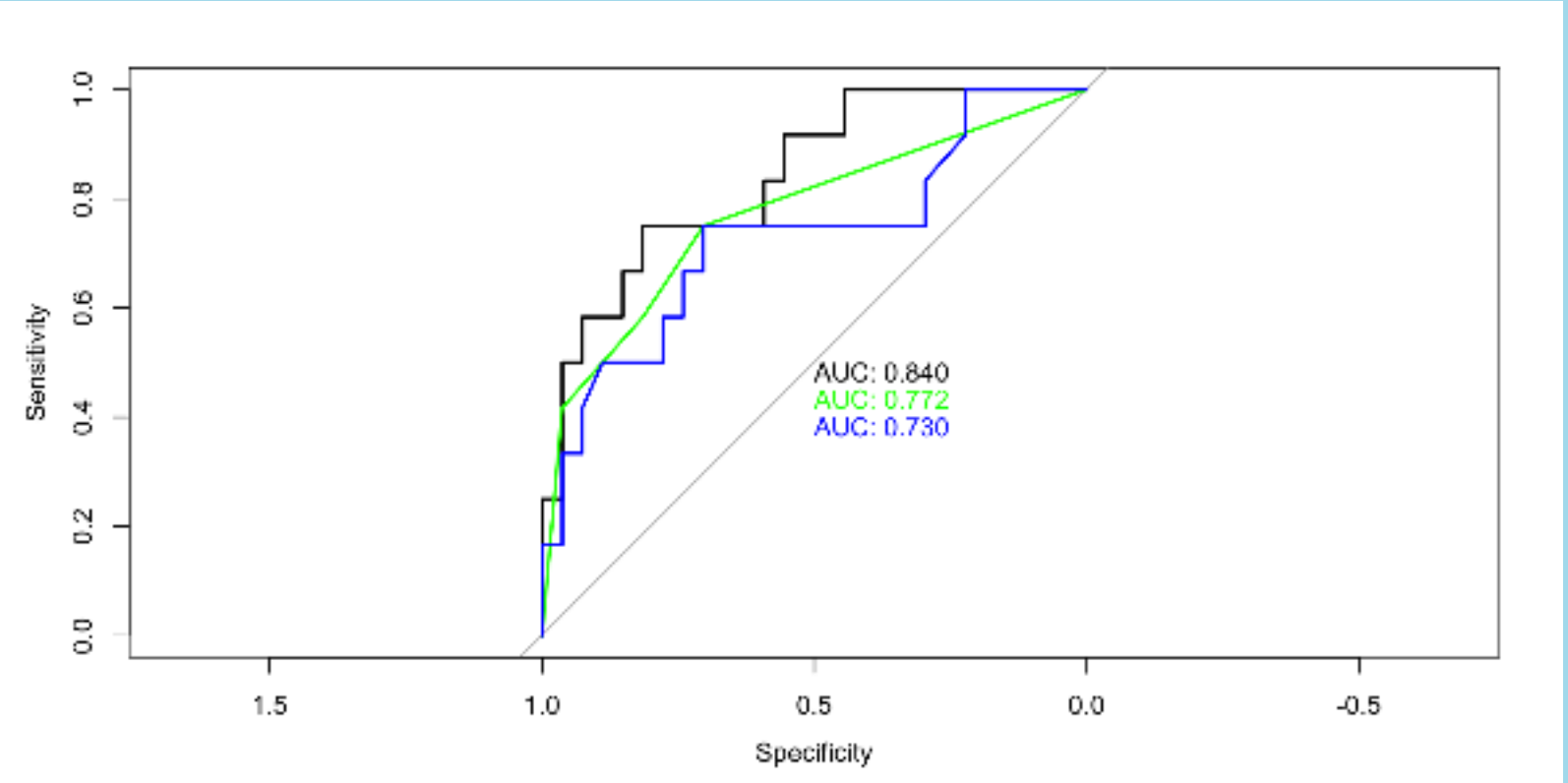
ROC in R (using pROC)

- 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

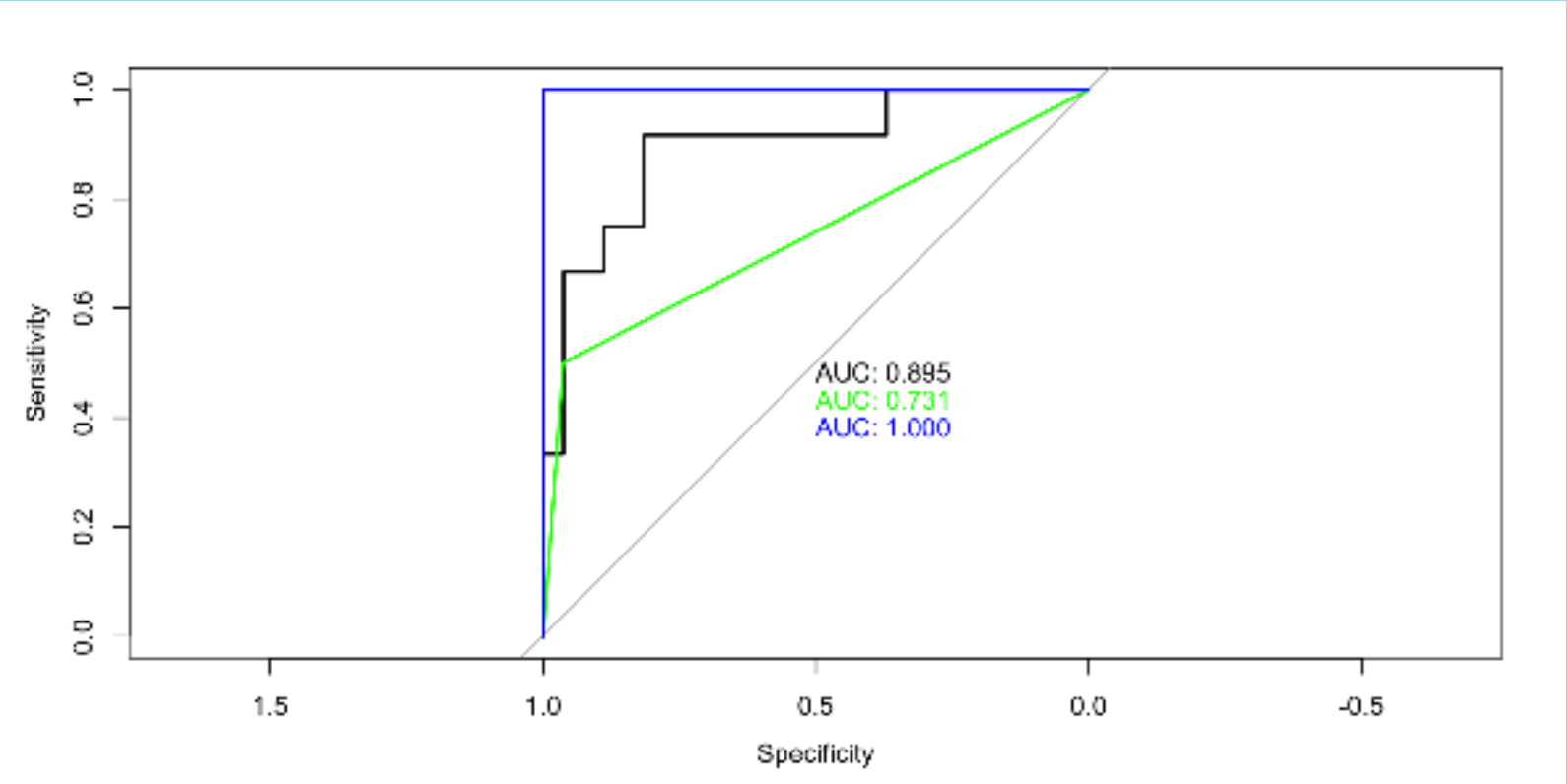
- Often want to make sure that fit/accuracy not a function of partition choice
- Cross-validation allows resampling of data (multiple times)
- K-fold - Data are split into K datasets of ~ equal size, model fit to $(K-1)(\frac{n}{K})$ observations to predict heldout set
- Leave One Out (LOO) - Model fit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

Crossvalidation in R using caret

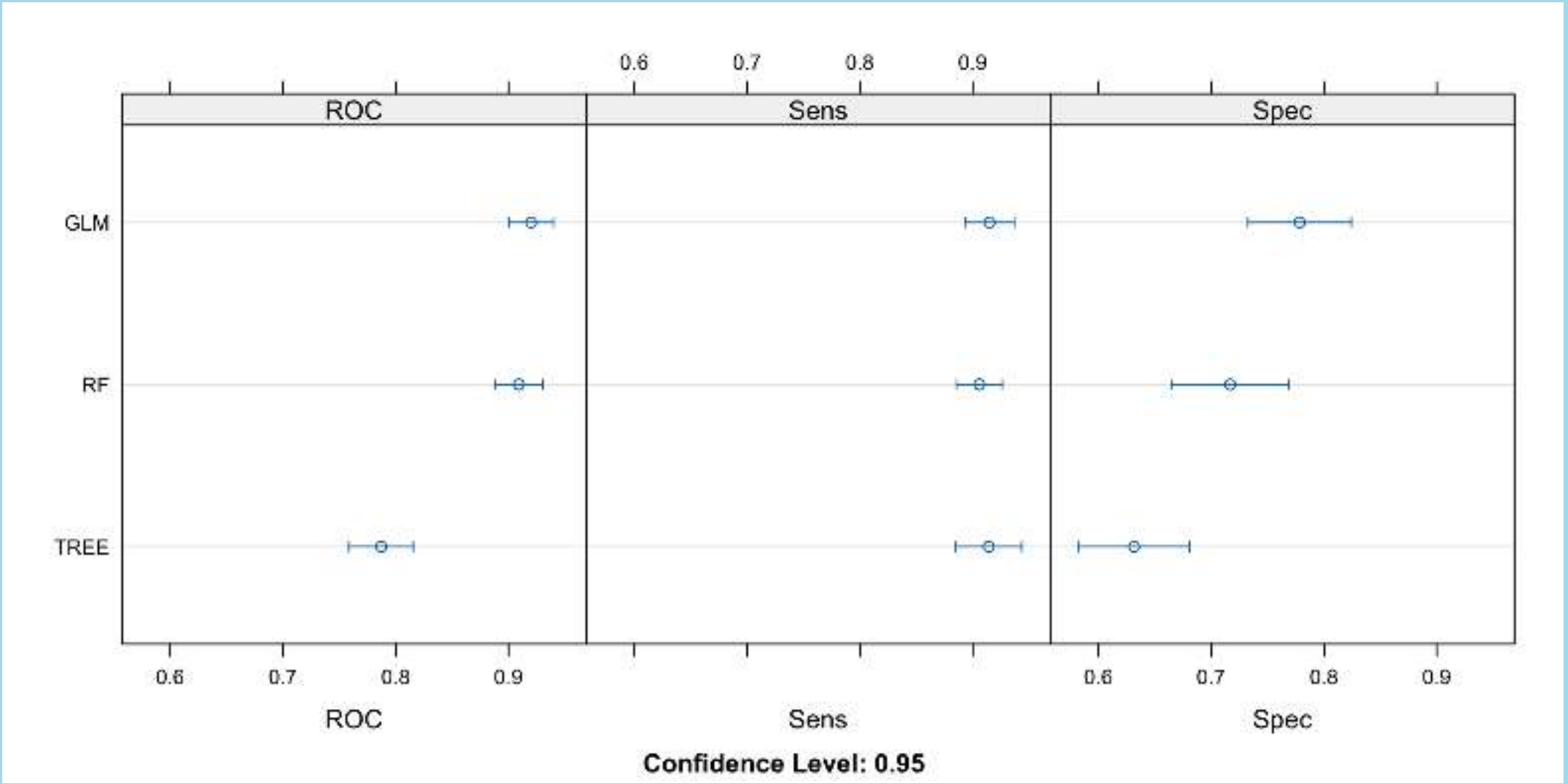
```
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 10,
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

log.model <- train(y ~., data = pts.df[,2:8],
                   method = "glm",
                   trControl = fitControl)
pred.log <- predict(log.model, newdata = testing[,2:8], type="prob")[,2]

tree.model <- train(y ~., data = pts.df[,2:8],
                   method = "rpart",
                   trControl = fitControl)

pred.tree <- predict(tree.model, newdata=testing[,2:8], type="prob")[,2]

rf.model <- train(y ~., 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"
```



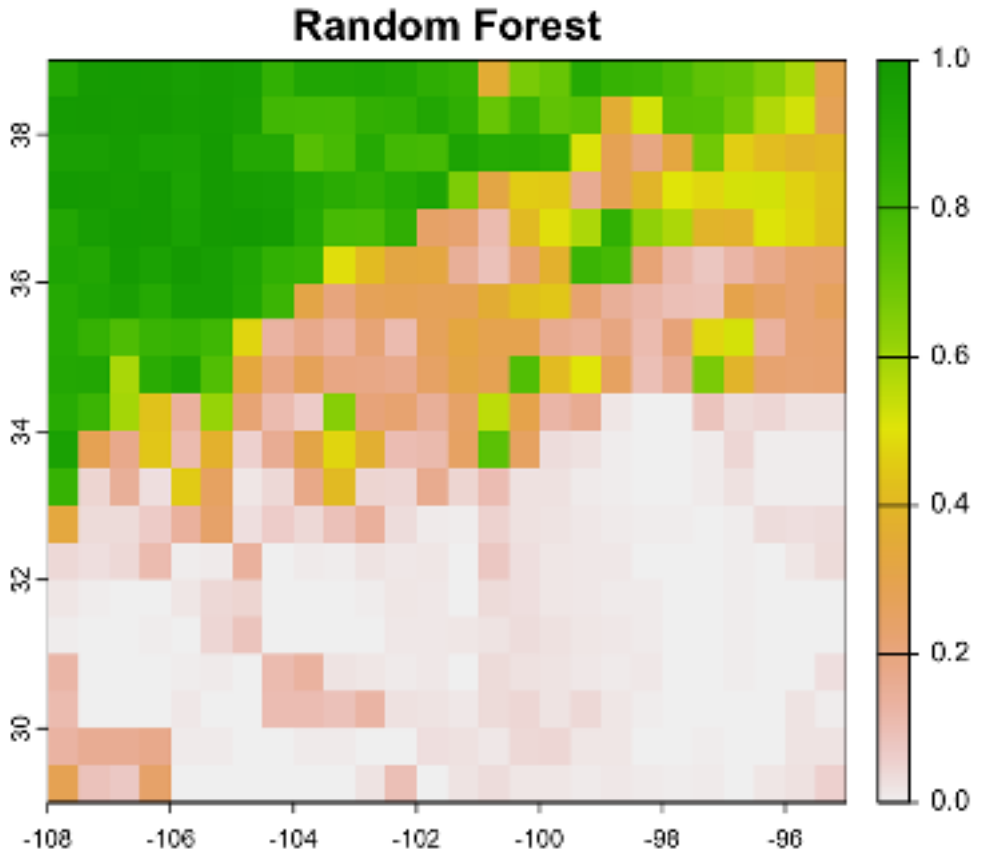
Crossvalidation in R using caret

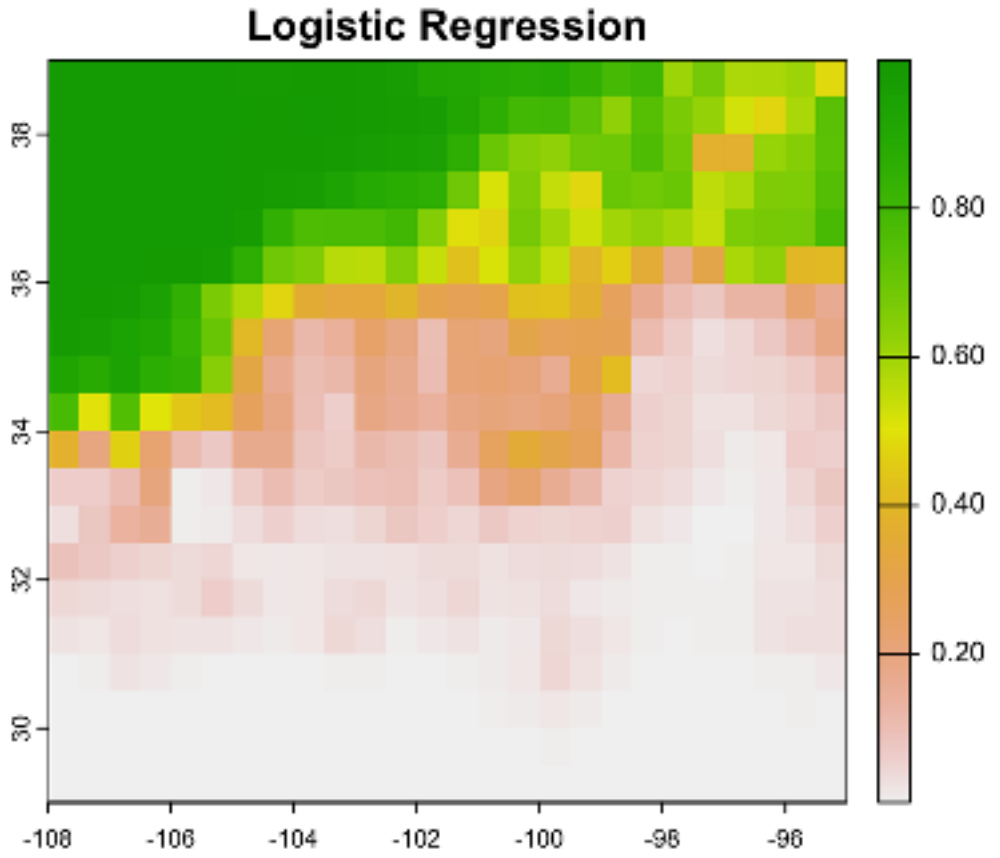


Spatial predictions

```
best.rf <- rf.model$finalModel
best.glm <- log.model$finalModel

rf.spatial <- terra::predict(pred.stack.scl, best.rf, type="prob")


glm.spatial <- terra::predict(pred.stack.scl, best.glm,type="response" )
```

Spatial predictions

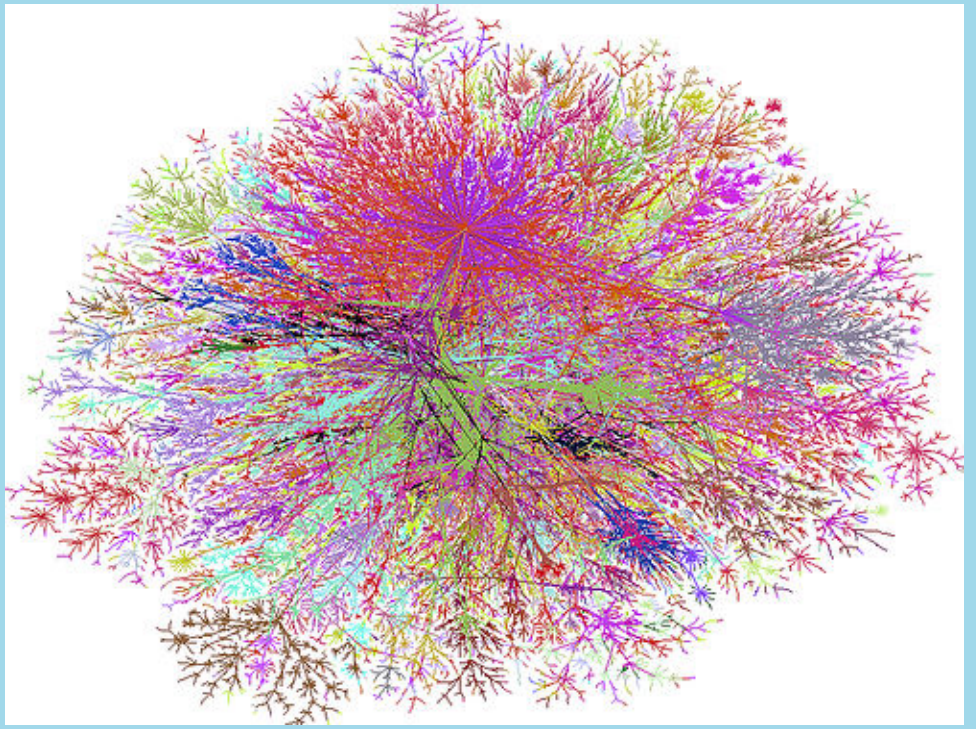




On to networks!

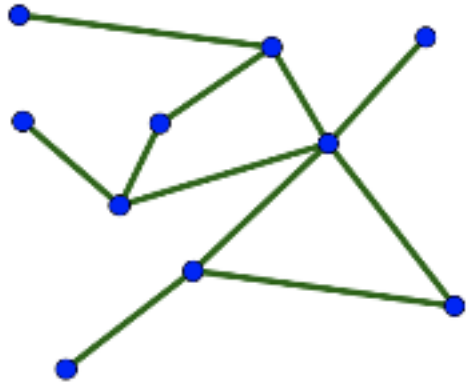
What is a network?

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

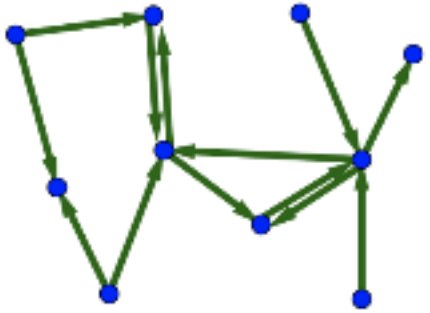


Types of networks

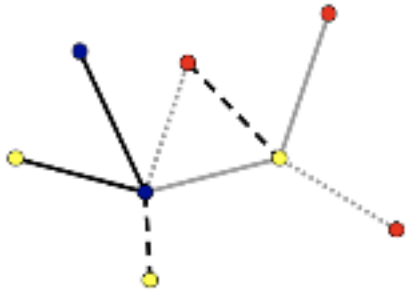
- (Un)directed
- Weighted
- Muli-type



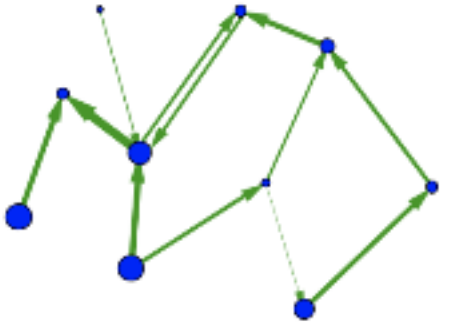
An undirected network.



A directed network.

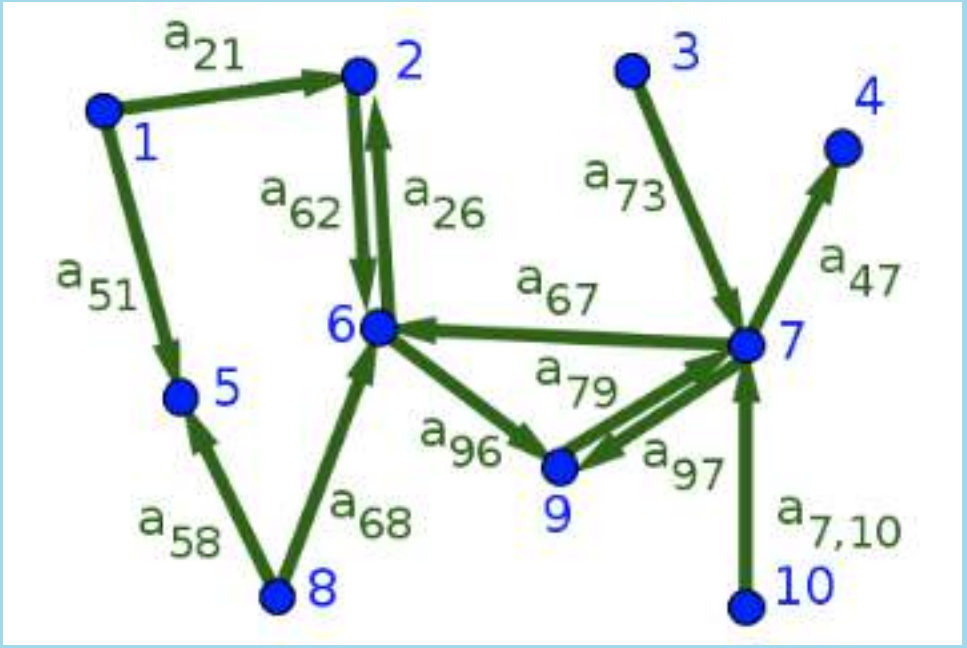


An undirected network where the nodes and edges have different types, as indicated by their colors and line styles.



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?