Slides sobre Machine Learning

James R. Hunter

UNIFESP/Sustentare

24-25 de agosto de 2018

Fonte

- Dr. Sharin Glander, Univ. de Münster, Alemanha
  - Webinar excelente recente
  - "Building meaningful machine learning models for disease prediction"
  - https://github.com/ShirinG
- Dados
  - UCI Machine Learning Repository
    - ★ U. de Wisconsin dados sobre câncer de mama
    - ★ Arquivo "breast-cancer-wisconsin-data.txt"

Machine Learning em Modelagem das Doenças

- Tipicamente, projetos com "big data"
- Modelo pode fornecer informação rapidamente e corretamente
  - Médicos podem usar a informação para desenhar tratamentos ou diagnósticos
- Aplicação para medicina personalizada de precisão
- Exemplo:
  - Diagnostico de câncer de mama com ajuda de modelo informatizado

Podemos Ter Confiança nos Modelos de Machine Learning?

- Algoritmos de ML modelam interações de alto grau enter as variáveis
- Interpretação dos resultados de ML pode ser difícil
- A "caixa preta" dos algoritmos de ML escondem como eles fazem escolhas
- Assim, *precisamos modelos que significam algo* para os
  - Arquitetos
  - Usadores
- "Meaningful Models"

O Que Faz um Modelo um "Meaningful Model"

- Poder generalizar baseado no modelo
- Responde à pergunta original
- . . . com suficiente precisão para ser confiável
- Grau de precisão depende no problema

Features – Covariáveis

- Variáveis para treinar o modelo
- Selecionar as variáveis certas – **crucial**
- Mais features não necessariamente bom
  - Perigo de "overfitting"

Vamos Pôr as Mãos na Massa

Dados

- Explicarei o modelo mas nós vamos focar na interpretação dos resultados
- Vêm de Wisconsin dados sobre câncer de mama
- Características dos tumores de mama
- Variável dependente: diagnose (`diag`)

Covariáveis - Caracteristicas dos Tumores

- Características
  - Sample ID (code number)
  - Clump thickness
  - Uniformity of cell size
  - Uniformity of cell shape
  - Marginal adhesion
  - Single epithelial cell size
  - Number of bare nuclei
  - Bland chromatin
  - Number of normal nuclei
  - Mitosis

Carregar Dados

```
bc_data <- read.table("~/Documents/Sustentare/Data_Analysis_with_R/dawR1/breast-cancer-wisconsin-data.txt",
                      header = FALSE,
                      sep = ",",
                      na.strings = "?")
colnames(bc_data) <- c("sample_code_number",
                       "clump_thickness",
                       "uniformity_of_cell_size",
                       "uniformity_of_cell_shape",
                       "marginal_adhesion",
                       "single_epithelial_cell_size",
                       "bare_nuclei",
                       "bland_chromatin",
                       "normal_nucleoli",
                       "mitosis",
                       "diag")

bc_data$diag <- ifelse(bc_data$diag == "2", "benigno",
                          ifelse(bc_data$diag == "4", "maligno", NA))
```

Dados

```
glimpse(bc_data)
```

```
## Observations: 699
## Variables: 11
## $ sample_code_number          <int> 1000025, 1002945, 1015425, 1016277...
## $ clump_thickness             <int> 5, 5, 3, 6, 4, 8, 1, 2, 2, 4, 1, 2...
## $ uniformity_of_cell_size     <int> 1, 4, 1, 8, 1, 10, 1, 1, 1, 2, 1, ...
## $ uniformity_of_cell_shape    <int> 1, 4, 1, 8, 1, 10, 1, 2, 1, 1, 1, ...
## $ marginal_adhesion           <int> 1, 5, 1, 1, 3, 8, 1, 1, 1, 1, 1, 1...
## $ single_epithelial_cell_size <int> 2, 7, 2, 3, 2, 7, 2, 2, 2, 2, 1, 2...
## $ bare_nuclei                 <int> 1, 10, 2, 4, 1, 10, 10, 1, 1, 1, 1...
## $ bland_chromatin             <int> 3, 3, 3, 3, 3, 9, 3, 3, 1, 2, 3, 2...
## $ normal_nucleoli             <int> 1, 2, 1, 7, 1, 7, 1, 1, 1, 1, 1, 1...
## $ mitosis                     <int> 1, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1...
## $ diag                        <chr> "benigno", "benigno", "benigno", "...
```

Analise de NAs – Decisão sobre o Que Fazer com Eles

- Quantas NAs estão nos dados?

```
length(which(is.na(bc_data)))
```

```
## [1] 16
```

- Quantas amostras perdemos se retiraram os NAs?

```
nrow(bc_data[is.na(bc_data), ])
```

```
## [1] 16
```

Imputar Valores de NAs

- Pacote e função `mice`
  - Multivariate Imputation by Chained Equations
- Cria dados imputados para dados incompletos multivariados
  - Gibbs Sampling (técnica Bayesiana)
  - Gera valores plausíveis sintéticos dado as outras colunas no dataset
- Imputação introduza mais incerteza no modelo

```r
summary(bc_data$bare_nuclei)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   1.000   1.000   3.545   6.000  10.000      16
```

```r
bc_data[,2:10] <- apply(bc_data[, 2:10], 2, function(x)
  X = as.numeric(as.character(x)))
dataset_impute <- mice(bc_data[, 2:10],  print = FALSE)
bc_data <- cbind(bc_data[, 11, drop = FALSE], mice::complete(dataset_impute, 1))
summary(bc_data$bare_nuclei)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.534   6.000  10.000
```

Resumo das Diagnoses

- Converter `diag` para um `factor`
- Quantos casos de benigno e maligna têm?

```
bc_data$diag <- as.factor(bc_data$diag)
 summary(bc_data$diag)
```

```
## benigno maligno
##     458     241
```

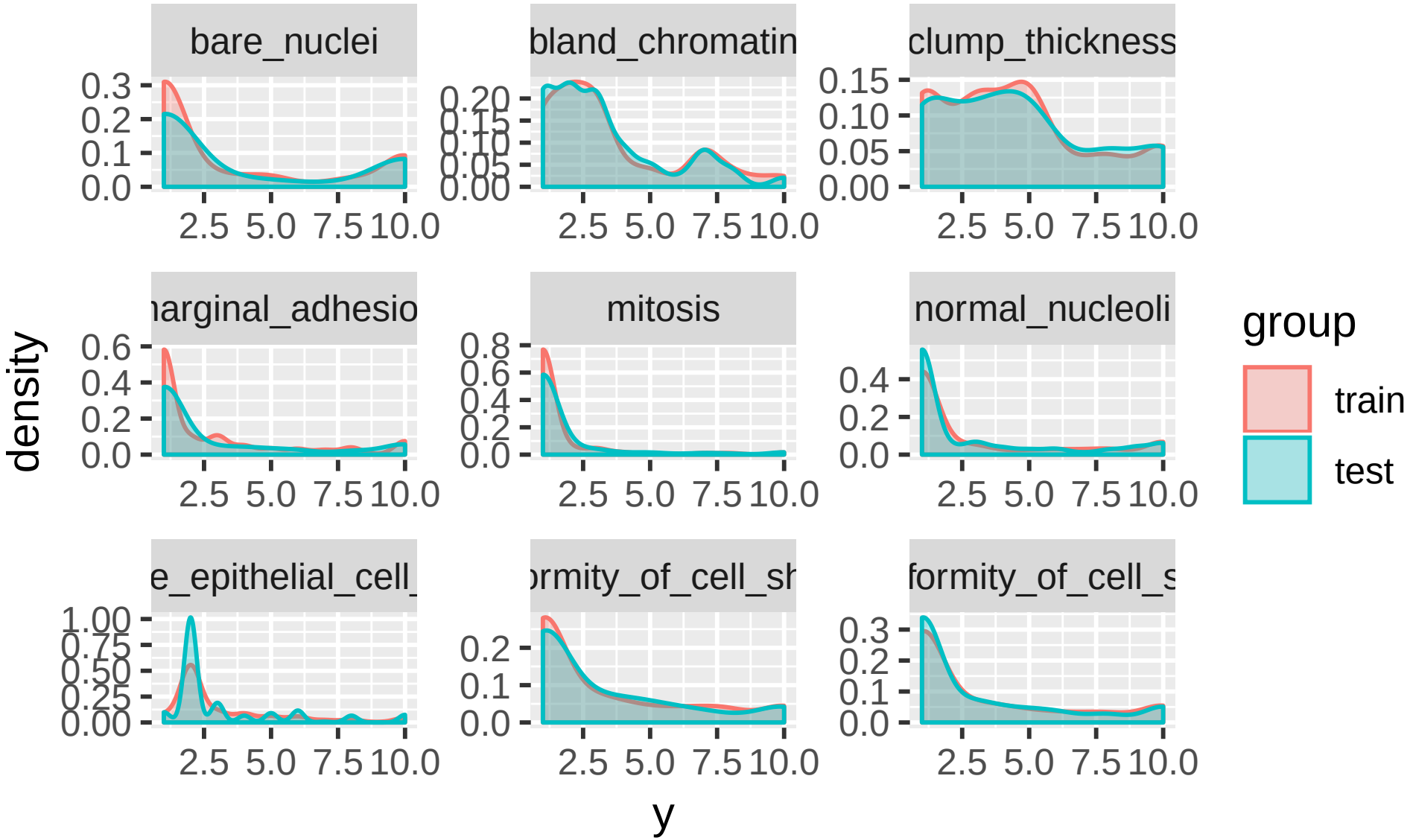
Classes de `diag` Desequilibradas

- Normalmente precisa um ajuste para tratar dessa desequilibração
- Não vamos fazer isso aqui

Criar as Bases Treinamento e Testes

```
set.seed(42)
indice <- createDataPartition(bc_data$diag, p = 0.7, list = FALSE)
train_data <- bc_data[indice, ] # use os índices para o treinamento
test_data <- bc_data[-indice, ] # use os outros para testes
```

as Bases Refletem os Mesmos Dados?



Exemplos dos Tipos de Modelos

- Regressão Linear
  - Ex: GLM
  - com `caret`
- Classificação com Arvores
  - Árvores recursivas de particionamento e regressão (pacote `rpart`)
  - Florestas Aleatórias ("Random Forests")
- Todos com `caret`

Controle de Treinamento

- Antes de iniciar o passo de treinar o modelos, precisamos decidir qual tipo de validação queremos usar
  - bootstrap, k-fold cross validation
- Especificar através da função `caret::trainControl()`
- Queremos usar *10-fold cross validation*
- Se pudermos repetir o processo de cross validation, faz a seleção do modelo ainda mais forte
  - Repetiremos 10 vezes

trainControl()

```
set.seed(42)
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 10,
                        savePredictions = TRUE,
                        verboseIter = FALSE)
```

Variável Dependente: *benign* ou *malignant*

- Qual tipo de análise mais relacionado?

Variável Dependente: *benign* ou *malignant*

- Qual tipo de análise mais relacionado?
- Regressão logistica

Treinamento do Modelo – Regressão Logistica

```
model_glm <- caret::train(diag ~ .,
                          data = train_data,
                          method = "glm",
                          preProcess = c("scale", "center"),
                          trControl = control)
```

Modelo

```
model_glm
```

```
## Generalized Linear Model
##
## 490 samples
##   9 predictor
##   2 classes: 'benigno', 'maligno'
##
## Pre-processing: scaled (9), centered (9)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 441, 441, 441, 441, 441, 441, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9600049  0.9106385
```

Resumo dos Resultados do Modelo

```
summary(model_glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2817  -0.1326  -0.0727   0.0262   2.4573
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -1.1554     0.3488  -3.312 0.000926 ***
## clump_thickness                1.3562     0.4253   3.189 0.001429 **
## uniformity_of_cell_size        0.1303     0.6611   0.197 0.843724
## uniformity_of_cell_shape       0.9738     0.7199   1.353 0.176150
## marginal_adhesion              0.9918     0.3695   2.684 0.007272 **
## single_epithelial_cell_size    0.2114     0.3720   0.568 0.569853
## bare_nuclei                    1.2119     0.3604   3.363 0.000772 ***
## bland_chromatin                1.1832     0.4716   2.509 0.012122 *
## normal_nucleoli                0.3365     0.3667   0.918 0.358857
## mitosis                        0.8190     0.5784   1.416 0.156746
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 631.346  on 489  degrees of freedom
## Residual deviance:  85.761  on 480  degrees of freedom
## AIC: 105.76
##
## Number of Fisher Scoring iterations: 8
```

O Modelo Pode Predizer os Resultados de Treinamento e de Teste?

- Função `predict()`
  - com modelo e valores para ser usados para previsão
- Aplicado a base de `train` como exemplo
- Mais interessante – base de `test`
  - Modelo nunca viu esses dados antes
- **Teste ácido**

Previsões

```
predtr <- predict(model_glm, train_data)
predtest <- predict(model_glm, test_data)
prop.table(table(predtest))
```

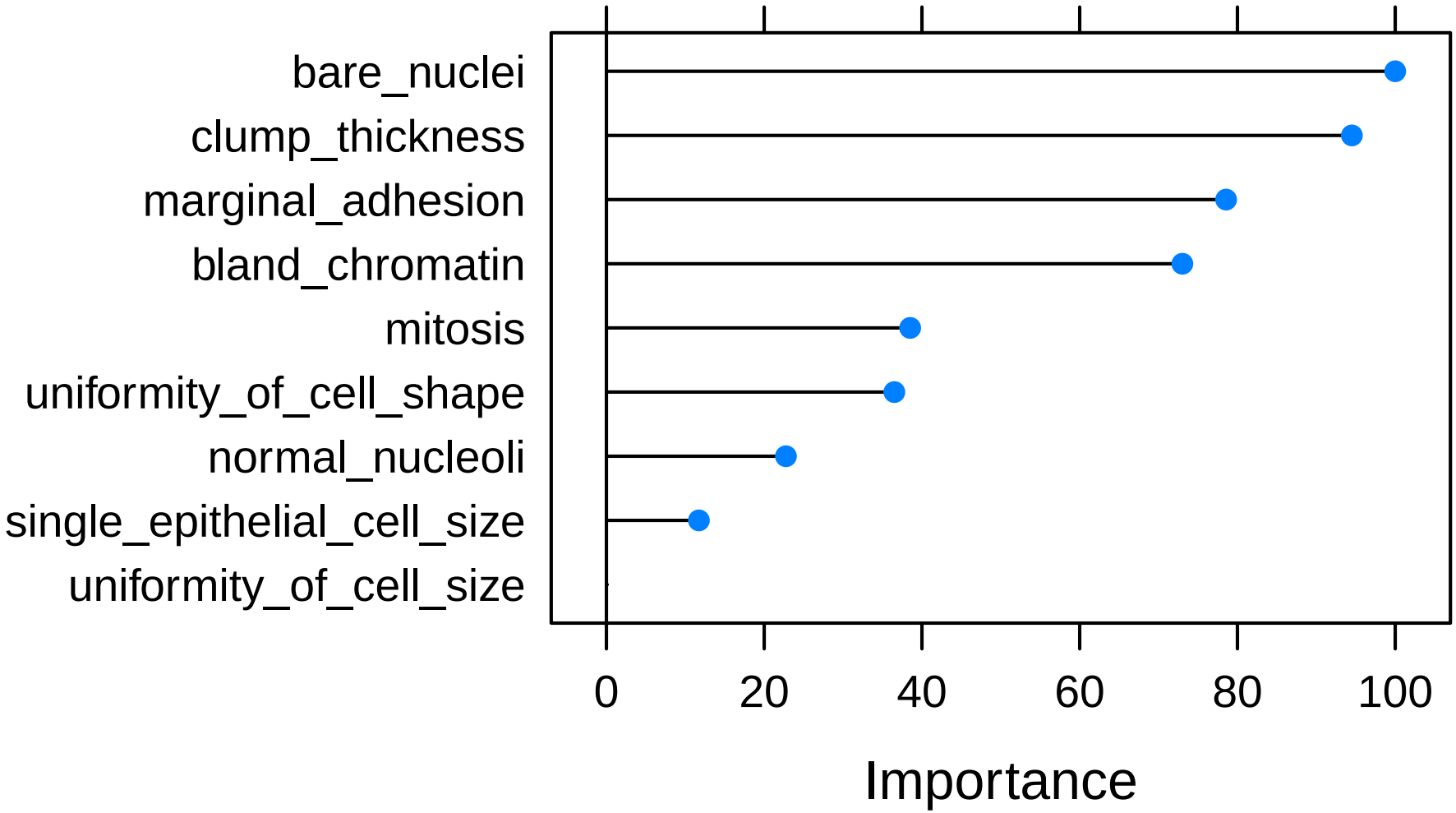
```
## predtest
##   benigno   maligno
## 0.6507177 0.3492823
```

```
prop.table(table(predtr))
```

```
## predtr
##   benigno   maligno
## 0.6510204 0.3489796
```

Quais Variáveis Têm Importância para o Modelo

```
plot(caret::varImp(model_glm))
```



Previsões com os Dados de Teste – Matriz de Confusão

```
confusionMatrix(predtest, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction benigno maligno
##    benigno     133       3
##    maligno       4      69
##
##                Accuracy : 0.9665
##                  95% CI : (0.9322, 0.9864)
##     No Information Rate : 0.6555
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9261
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9708
##             Specificity : 0.9583
##          Pos Pred Value : 0.9779
##          Neg Pred Value : 0.9452
##              Prevalence : 0.6555
##          Detection Rate : 0.6364
##    Detection Prevalence : 0.6507
##       Balanced Accuracy : 0.9646
##
##        'Positive' Class : benigno
##
```

Previsões com os Dados de Treinamento – Matriz de Confusão

```
confusionMatrix(predtr, train_data$diag)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction benigno maligno
##    benigno     313       6
##    maligno       8     163
##
##                Accuracy : 0.9714
##                  95% CI : (0.9525, 0.9843)
##     No Information Rate : 0.6551
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9369
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.9751
##             Specificity : 0.9645
##          Pos Pred Value : 0.9812
##          Neg Pred Value : 0.9532
##              Prevalence : 0.6551
##          Detection Rate : 0.6388
##    Detection Prevalence : 0.6510
##       Balanced Accuracy : 0.9698
##
##        'Positive' Class : benigno
##
```

"Receiver Operating Characteristic" (ROC) Validação do Modelo

- Desenvolvido ao início da WWII para determinar o que foi o sinal recebido pela nova tecnologia, *radar*
  - Avião ou pássaro
- Mede *sensibilidade* vs. *especificidade* de um modelo
- *Sensibilidade* = % do resultado positivo correto
  - Teste mede % dos resultados positivos das pessoas com uma doença
  - Taxa de previsões positivas certas ("True positive rate", TPR)
- *Especificidade* = % do resultado negativo correto
  - Teste mede % dos resultados negativos das pessoas sem uma doença
  - Taxa de previsões positivas erradas ("False positive rate", FPR)
  - Visualização da troca entre alta sensibilidade do modelo vs. alta especificidade
  - Não pode ter os 2 juntos

AUC (Área abaixo da Curva)

- AUC mede quanto porcentagem da área do gráfico a curva do modelo cobre
- 100% quer dizer que o modelo é perfeitamente sensível e especifico
- 50% quer dizer que o resultado é puramente aleatório
- Modelos com AUC maiores prevêm melhor que eles com AUC menores
- Pergunta:
  - Como calcular área abaixo de uma curva qualquer em matemática?

ROC em R

- 2 Pacotes
  - `pROC`
  - `ROCR`
- Iguais (basicamente)
- Começamos com `pROC`
  - Comando principal – `roc`

pROC::roc()

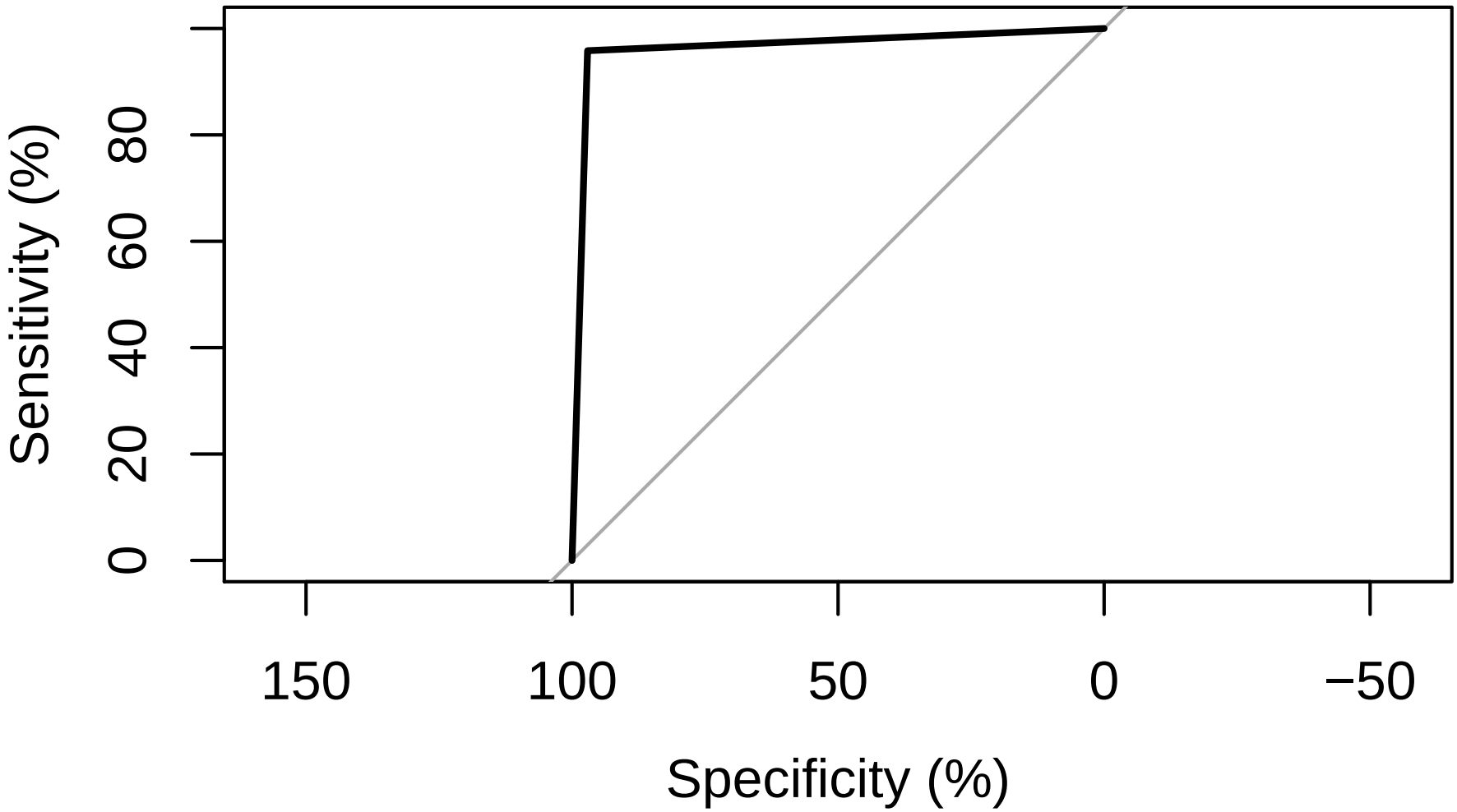
- Compara as previsões contra as observações
- Previsões precisam ser numéricas (não `factor`)
- Use as opções seguintes:
  - `plot = TRUE, percent = TRUE, ci = TRUE, grid = TRUE`
- Produz um gráfico e dados sobre o AUC

Chamada e Estatísticas

```
## colocar predtest na faixa de 0:1 (atualmente 1:2)
predtestroc <- as.numeric(predtest) - 1 # para curva ROC números devem ser 1 e 0
rocteste <- roc(response = test_data$diag,
                predictor = predtestroc,
                levels = c("benigno", "maligno"),
                plot = FALSE, percent = TRUE,
                ci = TRUE, grid = TRUE)
rocteste
```

```
##
## Call:
## roc.default(response = test_data$diag, predictor = predtestroc,     levels = c("
##
## Data: predtestroc in 137 controls (test_data$diag benigno) < 72 cases (test_data
## Area under the curve: 96.46%
## 95% CI: 93.74%-99.18% (DeLong)
```

Gráfico



Procedimento com ROCR

- ROCR quer os dados num formato específico
  - Precisa refazer a previsão utilizando a função deste pacote
  - Função usará uma versão numérica das previsões predtest
  - Depois calcular os valores da curva e fazer o gráfico
  - ROCR utiliza a terminologia "tpr" e "fpr" para gráfico ROC
  - Pode imprimir sensibilidade e especificidade com sens, spec

```
## Fazer previsão do modelo com ROCR
ROCRpred <- prediction(as.numeric(predtestroc), test_data$diag)
ROCRperf <- performance(ROCRpred, "tpr", "fpr")
```

Gráfico

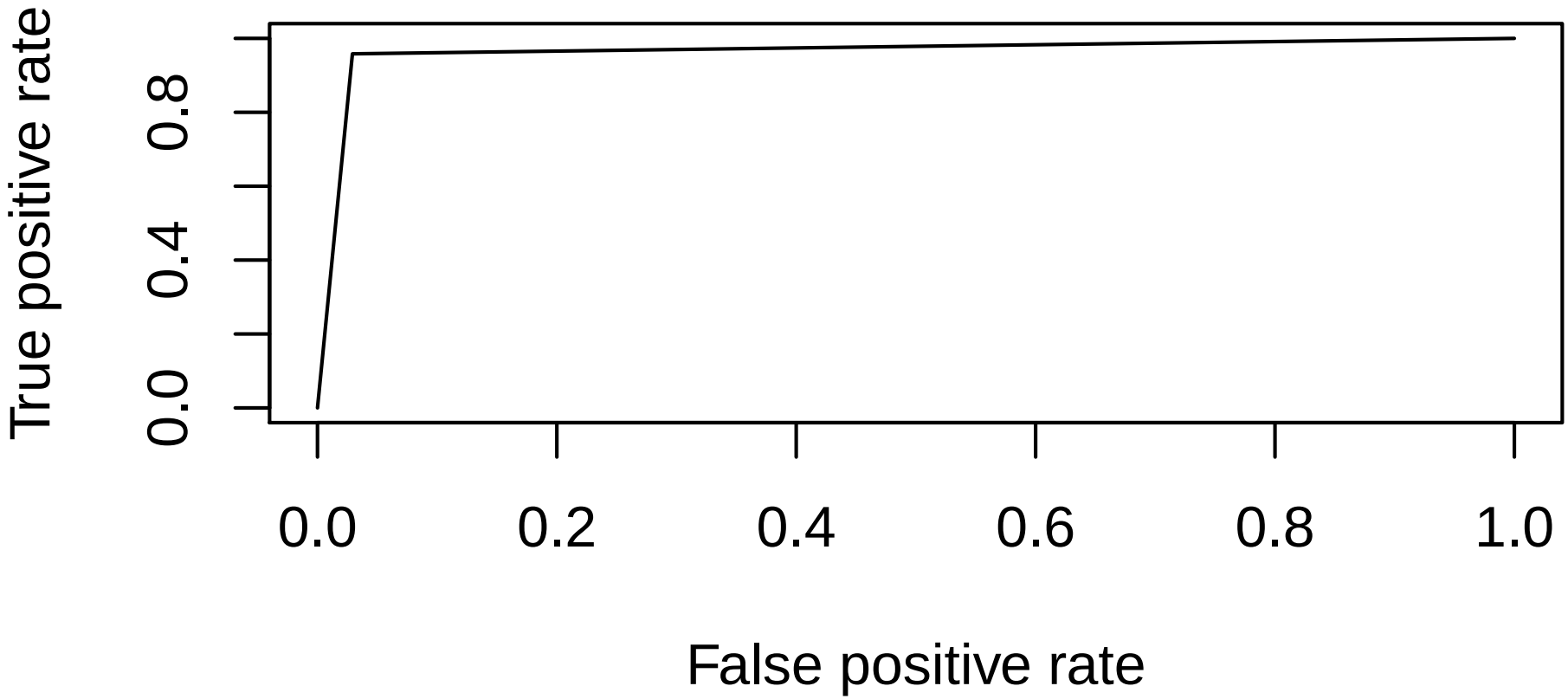
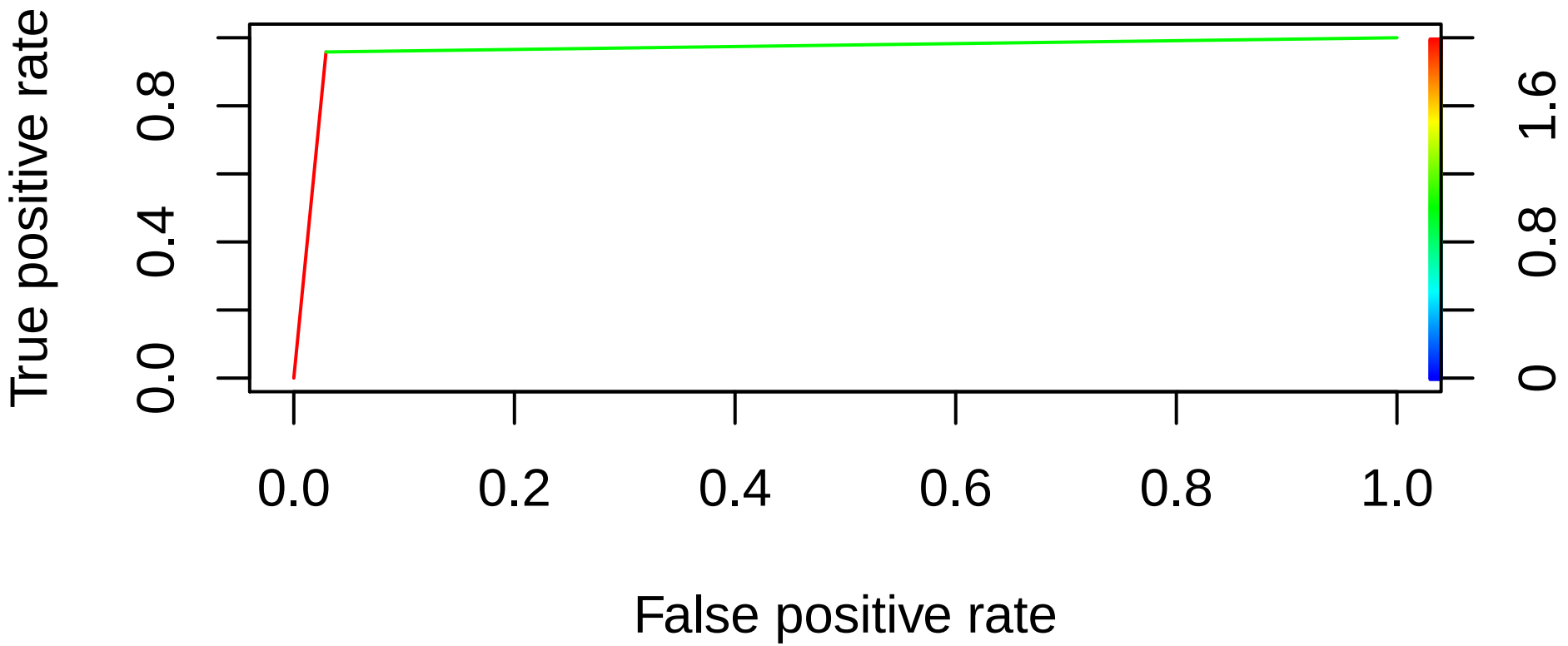

True positive rate
0.0
0.4
0.8
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

Gráfico com Cores

```
plot(ROCRperf, colorize = TRUE)
```



Limites da Decisão sobre `diag`

- Onde no gráfico fica a troca ótima?
  - No ponto mais para cima e para esquerda
- `pROC::coords()` pode calcular este ponto
- Precisa dar as seguintes informações a função:
  - nome de objeto de ROC
  - Palavra "best"
  - Coordenados para retornar a você ("threshold")

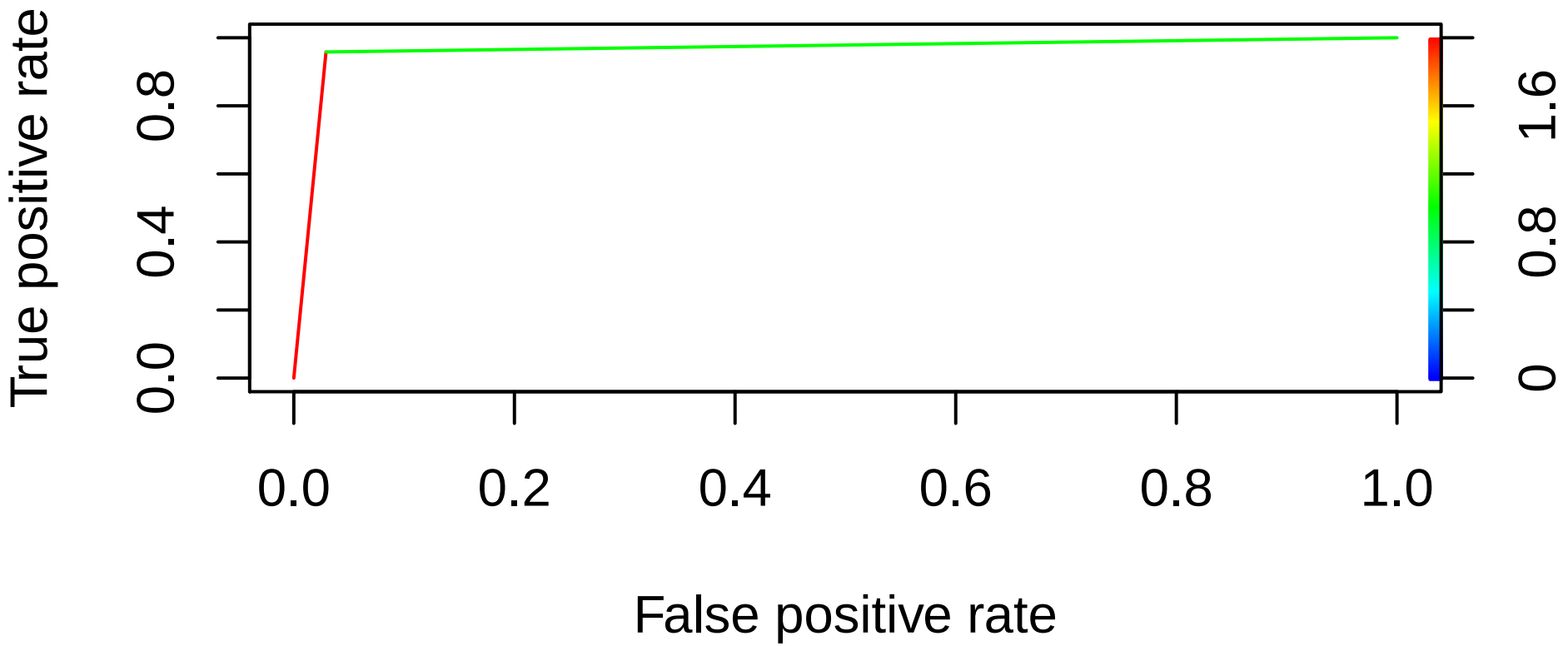
Limites de Nosso Modelo

```
coords(rocteste, "best", ret = "threshold")
```

```
## [1] 0.5
```

Gráfico com Cores

```
plot(ROCRperf, colorize = TRUE)
```



Novo Modelo – Modelos de Arvore – `rpart`

- Modelos que constroem arvores de decisão
- Excelentes para problemas de classificação
- Pacote `rpart`
- Gráficos mostra como escolha das classes está sendo feita
  - Gráfico vem do pacote `rpart.plot`

Como Funciona uma Arvore

- Cf. Kuhn & Johnson, *Applied Predictive Modeling* (2013)
- Feita de *nodos* e *ramos*
- Ramos conectam nodos até que chegar num nodo terminal
- Algoritmo cria uma serie de partilhas (divisões) baseado em testes lógicos aninhados
- Os testes lógicos definem a previsão que o modelo faria com novos dados

Exemplo de uma Regra de uma Arvore

```
if Predictor A >= 1.7 then
|   if Predictor B >= 202.1 then Outcome = 1.3
|   else Outcome = 5.6
else Outcome 2.5
```

Arvores São uma Técnica de Machine Learning Popular

- Interpretação fácil
- Podem lidar com muitas convariáveis de vários tipos
- Não precisa descrever exatamente a relação entre
  - Variável dependente
  - Variáveis independentes
- NA's não criam problemas
- Mas, tem desvantagens também
  - São instáveis (pequena mudança numa variável pode cause grande mudança no resultado)
  - Exatidão de previsões não tão boa que outros tipos de modelos

Funcionamento do Modelo de Arvore

- Algoritmo divide os dados em grupos menores que são mais homogêneos com a dependente
- 3 Critérios para divisão
  - Qual variável de previsão para usar para o "split"
  - Profundidade da arvore
  - A equação de previsão nos nodos terminais
- Metodologia de `rpart` vem de Breiman et. al (1984)
  - Classification and regression tree (CART)

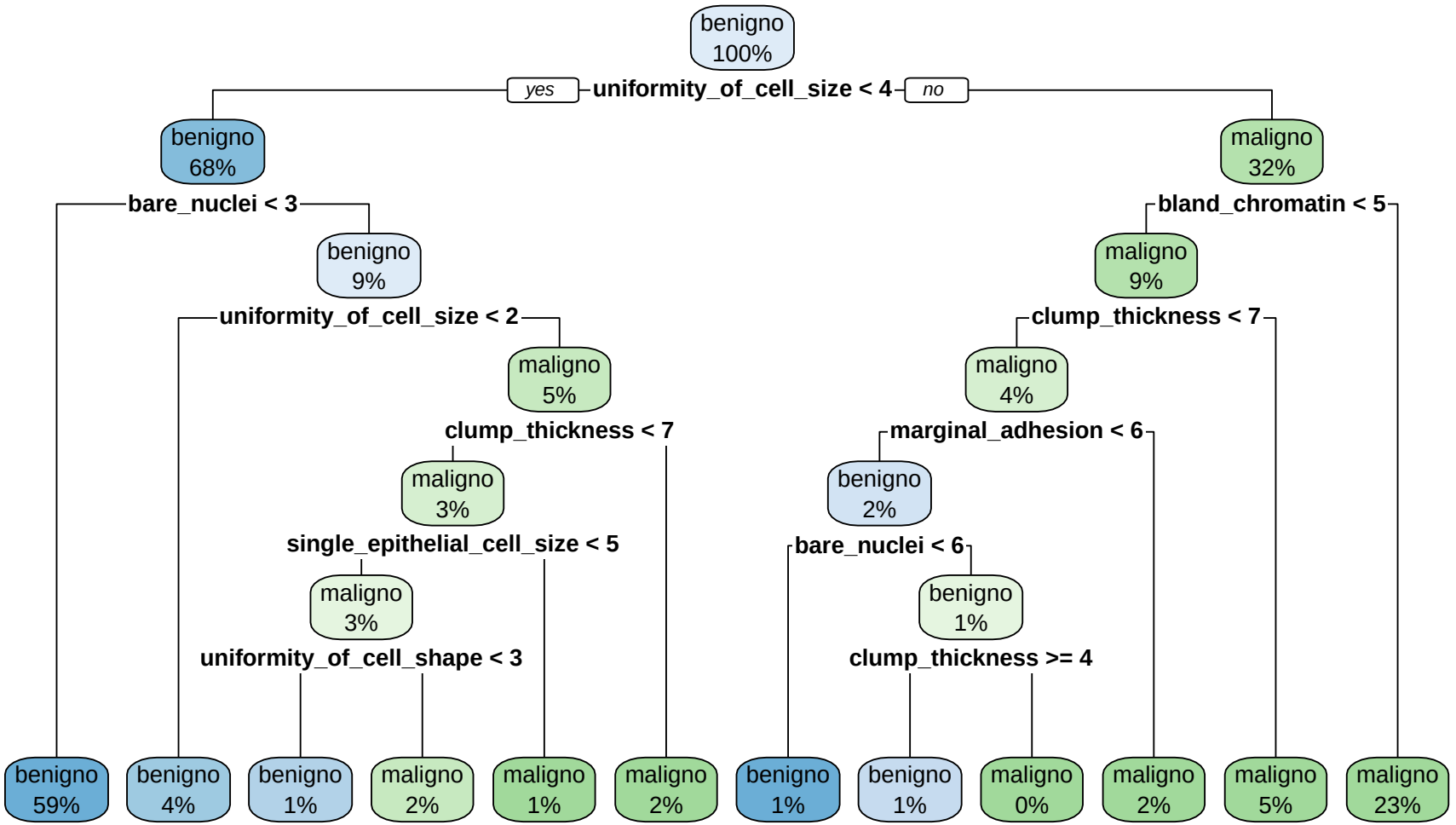
Paramétros Chaves para `rpart`

- `method`
  - Para classificação: “class”
  - Para regressão: “anova”
- `control`
  - Vai chamar `rpart.control` explicito
  - `xval`: número de cross-validations
  - `minbucket`: número mínimo de observações em um nodo terminal
- `parms` – parâmetros para dividindo os casos
  - Só usado para classificação
  - `information`

Vamos Construir Um Modelo de Câncer de Mama

```
set.seed(42)
fitree1 <- rpart(diag ~ .,
            data = train_data,
            method = "class",
            control = rpart.control(xval = 10,
                                    minbucket = 2,
                                    cp = 0),
             parms = list(split = "information"))
```

Arvore


benigno
100%
yes
uniformity_of_cell_size < 4
no
benigno
68%
maligno
32%
bare_nuclei < 3
bland_chromatin < 5
benigno
9%
maligno
9%
uniformity_of_cell_size < 2
clump_thickness < 7
maligno
5%
maligno
4%
clump_thickness < 7
marginal_adhesion < 6
maligno
3%
benigno
2%
single_epithelial_cell_size < 5
bare_nuclei < 6
maligno
3%
benigno
1%
uniformity_of_cell_shape < 3
clump_thickness >= 4
benigno
59%
benigno
4%
benigno
1%
maligno
2%
maligno
1%
maligno
2%
benigno
1%
benigno
1%
maligno
0%
maligno
2%
maligno
5%
maligno
23%

Resumo do Modelo de rpart

```
summary(fitree1, cp = 1)
```

```
## Call:
## rpart(formula = diag ~ ., data = train_data, method = "class",
##     parms = list(split = "information"), control = rpart.control(xval = 10,
##         minbucket = 2, cp = 0))
##   n= 490
##
##            CP nsplit  rel error    xerror       xstd
## 1 0.822485207      0 1.00000000 1.0000000 0.06226029
## 2 0.038461538      1 0.17751479 0.1775148 0.03140182
## 3 0.009861933      3 0.10059172 0.1360947 0.02770369
## 4 0.005917160      6 0.07100592 0.1597633 0.02988737
## 5 0.000000000     11 0.04142012 0.1360947 0.02770369
##
## Variable importance
##     uniformity_of_cell_size                 bare_nuclei
##                          20                          17
##    uniformity_of_cell_shape             bland_chromatin
##                          17                          14
##             normal_nucleoli single_epithelial_cell_size
##                          14                          13
##             clump_thickness                     mitosis
##                           2                           1
##           marginal_adhesion
##                           1
##
## Node number 1: 490 observations
##   predicted class=benigno  expected loss=0.344898  P(node) =1
##     class counts:   321   169
##    probabilities: 0.655 0.345
```

Previsões com a Arvore

```
predtest <- predict(fitree1, newdata = test_data, type = "class")
prop.table(table(predtest))
```

```
## predtest
##   benigno   maligno
## 0.6698565 0.3301435
```

Confusion Matrix – Arvore

```
confusionMatrix(predtest, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction benigno maligno
##    benigno     133       7
##    maligno       4      65
##
##                Accuracy : 0.9474
##                  95% CI : (0.9078, 0.9734)
##     No Information Rate : 0.6555
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8823
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.9708
##             Specificity : 0.9028
##          Pos Pred Value : 0.9500
##          Neg Pred Value : 0.9420
##              Prevalence : 0.6555
##          Detection Rate : 0.6364
##    Detection Prevalence : 0.6699
##       Balanced Accuracy : 0.9368
##
##        'Positive' Class : benigno
##
```

ROC Dados

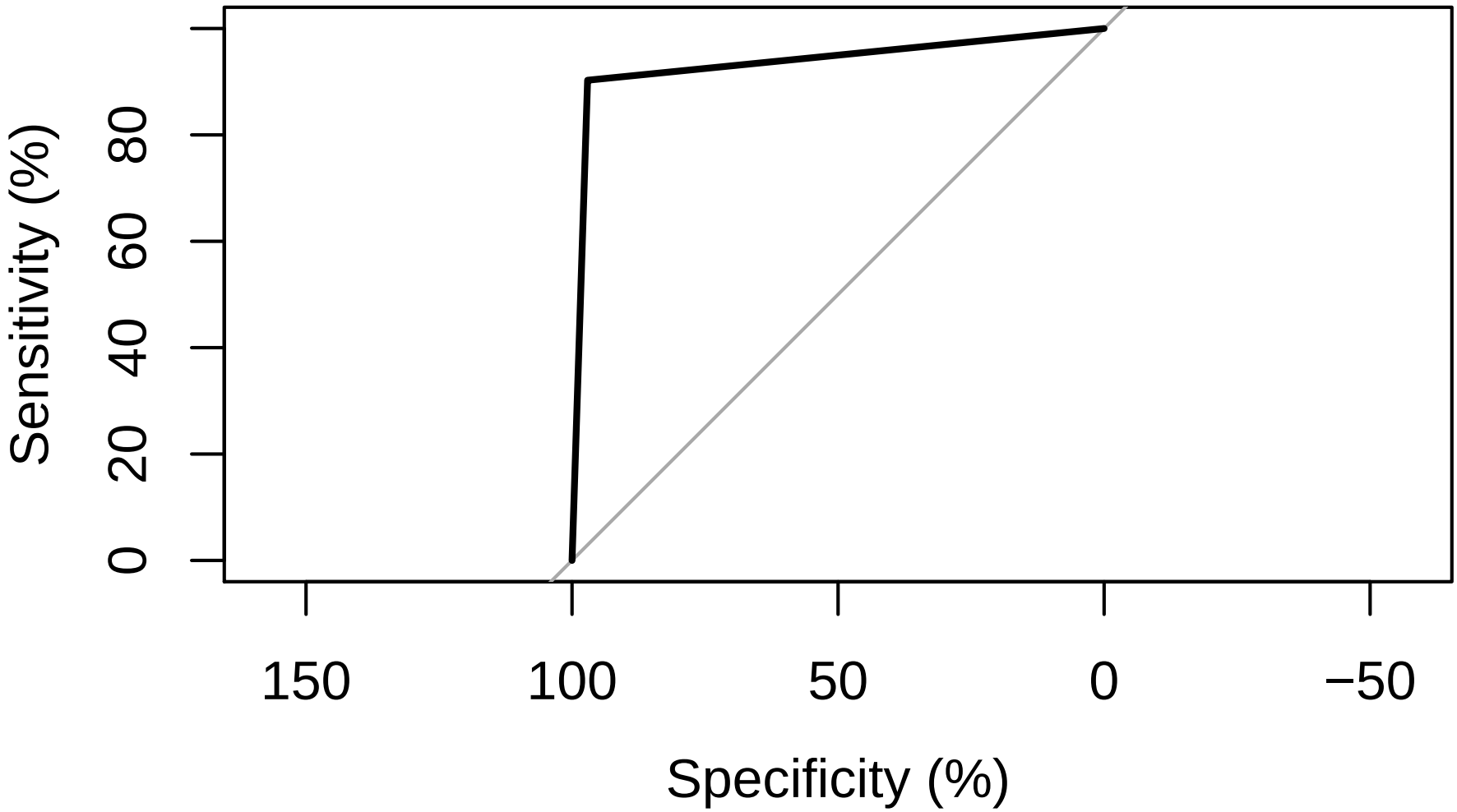
```
## colocar predtest na faixa de 0:1 (atualmente 1:2)
predtestroc <- as.numeric(predtest) -1
rocteste <- roc(response = test_data$diag,
                predictor = predtestroc,
                levels = c("benigno", "maligno"),
                plot = FALSE, percent = TRUE,
                ci = TRUE, grid = TRUE)
rocteste
```

```
##
## Call:
## roc.default(response = test_data$diag, predictor = predtestroc,     levels = c("
##
## Data: predtestroc in 137 controls (test_data$diag benigno) < 72 cases (test_data
## Area under the curve: 93.68%
## 95% CI: 89.95%-97.4% (DeLong)
```

```
suppressMessages(coords(rocteste, "best", ret = "threshold"))
```

```
## [1] 0.5
```

Gráfico



Arvores Mais Robustas – Random Forests

- Random Forests elaborado como algoritmo por Breiman em 2000
- Ideia básica: Combinando resultados de muitas arvores vai produzir uma arvore final melhor

  Grow many deep regression trees to randomized versions of the training data, and average them. Efron & Hastie, 2016

- "Randomized versions" – pode ser bootstrapping ou outras técnicas de re-amostragem

Random Forests em R

- Pacote `randomForest`
- Formato:

```
randomForest(y ~ xvars, data = dados, ntrees = 1000,
             importance = TRUE)
```

- `y` deve ser expressa como `factor` para classificação
- Argumentos chaves:
  - `ntrees`: número de arvores para a calcular; deve ser muito maior que o número das covariáveis
  - `importance = TRUE`: para calcular os valores para importância dos variáveis

Random Forests Aplicado ao Câncer de Mama

```
arvores = 100
rffit <- randomForest(as.factor(diag) ~ ., data = train_data,
                      ntree = arvores, importance = TRUE, proximity = TRUE)
rffit
```

```
##
## Call:
##  randomForest(formula = as.factor(diag) ~ ., data = train_data,      ntree = arv
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 3.27%
## Confusion matrix:
##         benigno maligno class.error
## benigno     312       9  0.02803738
## maligno       7     162  0.04142012
```

Confusion Matrix aqui é dos dados de *treinamento*

OOB Error????

- "Out of Bag"
  - Para todos as arvores, os erros associados com os valores não utilizados no treinamento do modelo
  - Como fizemos com cross-validation

Previsões com a Random Forest

```
predtest <- predict(rffit, newdata = test_data, type = "class")
prop.table(table(predtest))
```

```
## predtest
##  benigno  maligno
## 0.645933 0.354067
```

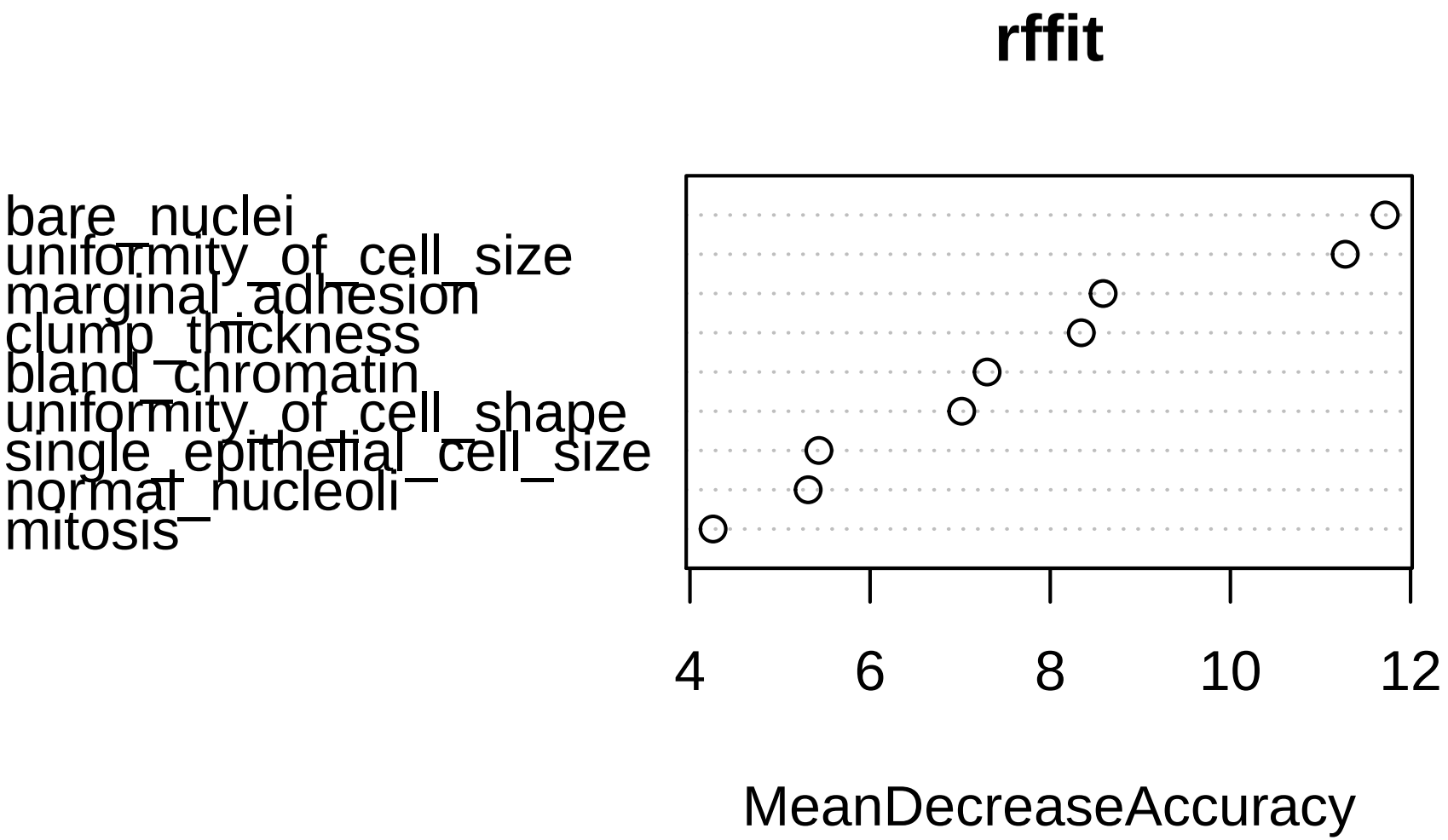
Desempenho de Random Forest

```
confusionMatrix(predtest, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction benigno maligno
##    benigno     133       2
##    maligno       4      70
##
##                Accuracy : 0.9713
##                  95% CI : (0.9386, 0.9894)
##     No Information Rate : 0.6555
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9369
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9708
##             Specificity : 0.9722
##          Pos Pred Value : 0.9852
##          Neg Pred Value : 0.9459
##              Prevalence : 0.6555
##          Detection Rate : 0.6364
##    Detection Prevalence : 0.6459
##       Balanced Accuracy : 0.9715
##
##        'Positive' Class : benigno
##
```

Importância das Variáveis

```
randomForest::varImpPlot(rffit, type = 1)  ## NB, função dentro de randomForest
```



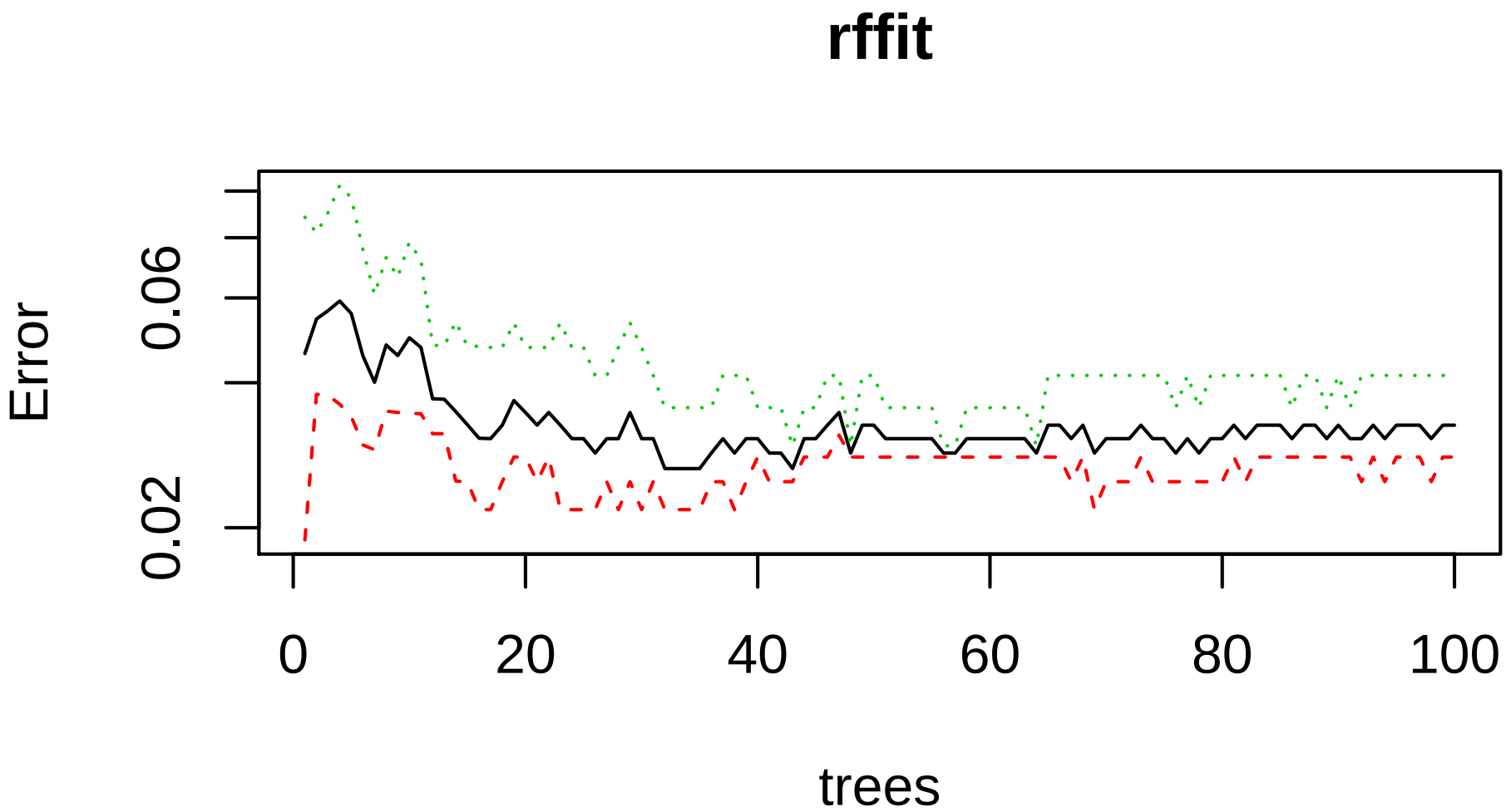
O Que Quer Dizer "Mean Decrease Accuracy"

- Através de todos as arvores - A variável causa uma perda de precisão no modelo
- Variáveis que podem causar perda de precisõ são mais importantes
- Exemplos:
  - "bare nuclei" é a mais importante porque pode causar mais perda
  - "mitosis" é o menos importante, porque qualquer valor que assuma não vai afetar o resultado do modelo, `diag`

Controle de Erros

- Gráfico de redução de MSE com o número de arvores calculadas

```
plot(rffit, log = 'y')
```



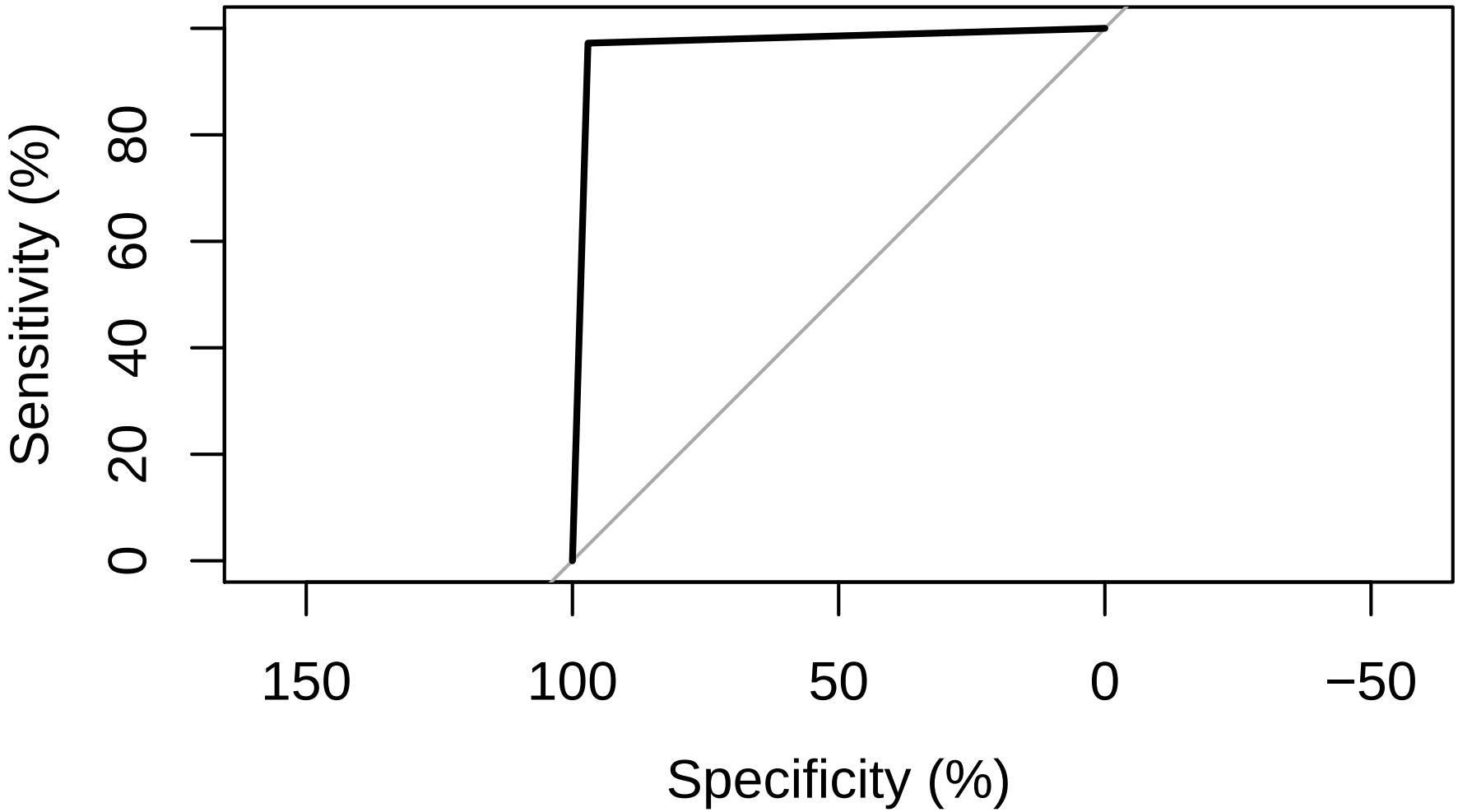
Curva ROC e AUC para Random Forests

```
## colocar predtest na faixa de 0:1 (atualmente 1:2)
predtestroc <- as.numeric(predtest) - 1
rocteste <- roc(response = test_data$diag,
                predictor = predtestroc,
                levels = c("benigno", "maligno"),
                plot = FALSE, percent = TRUE,
                ci = TRUE, grid = TRUE)
rocteste
```

```
##
## Call:
## roc.default(response = test_data$diag, predictor = predtestroc,     levels = c("
##
## Data: predtestroc in 137 controls (test_data$diag benigno) < 72 cases (test_data
## Area under the curve: 97.15%
## 95% CI: 94.77%-99.53% (DeLong)
```

```
suppressMessages(coords(rocteste, "best", ret = "threshold"))
```

```
## [1] 0.5
```


Sensitivity (%)
0
20
40
60
80
150
100
50
0
−50
Specificity (%)

Fazer Random Forests com `caret`

- Só precisa mudar o a especificação de `train`
- `method = "rf"`
- `caret` chama `randomForest` para fazer os calculos
  - *wrapper* função
- Aqui vamos fazer `set.seed(42)` para ser consistente com os outros métodos

Calcular os Random Forests

```
set.seed(42)
model_rf <- caret::train(diag ~ .,
                         data = train_data,
                         method = "rf",
                         preProcess = c("scale", "center"),
                         trControl = control)
```

Resultados Básicos – RF – caret

```
model_rf
```

```
## Random Forest
##
## 490 samples
##   9 predictor
##   2 classes: 'benigno', 'maligno'
##
## Pre-processing: scaled (9), centered (9)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 441, 441, 441, 441, 441, 441, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9710597  0.9361001
##   5     0.9645202  0.9211996
##   9     0.9612418  0.9137172
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Calcular as Variáveis Importantes

```
imp <- model_rf$finalModel$importance # Guarda em unidades originais
importance <- varImp(model_rf, scale = TRUE) # Scale coloca em escala de 100 -> 0
```

Variáveis Importantes – Escala Original

- % das arvores em que a variável aparece (eu acho??)

```
imp[order(imp, decreasing = TRUE), ]
```

```
##     uniformity_of_cell_size    uniformity_of_cell_shape
##                   49.934229                   40.077846
##                bare_nuclei                bland_chromatin
##                   32.608649                   25.310486
##             normal_nucleoli single_epithelial_cell_size
##                   22.202463                   20.182933
##             clump_thickness           marginal_adhesion
##                   14.788615                   12.057022
##                     mitosis
##                    2.444551
```

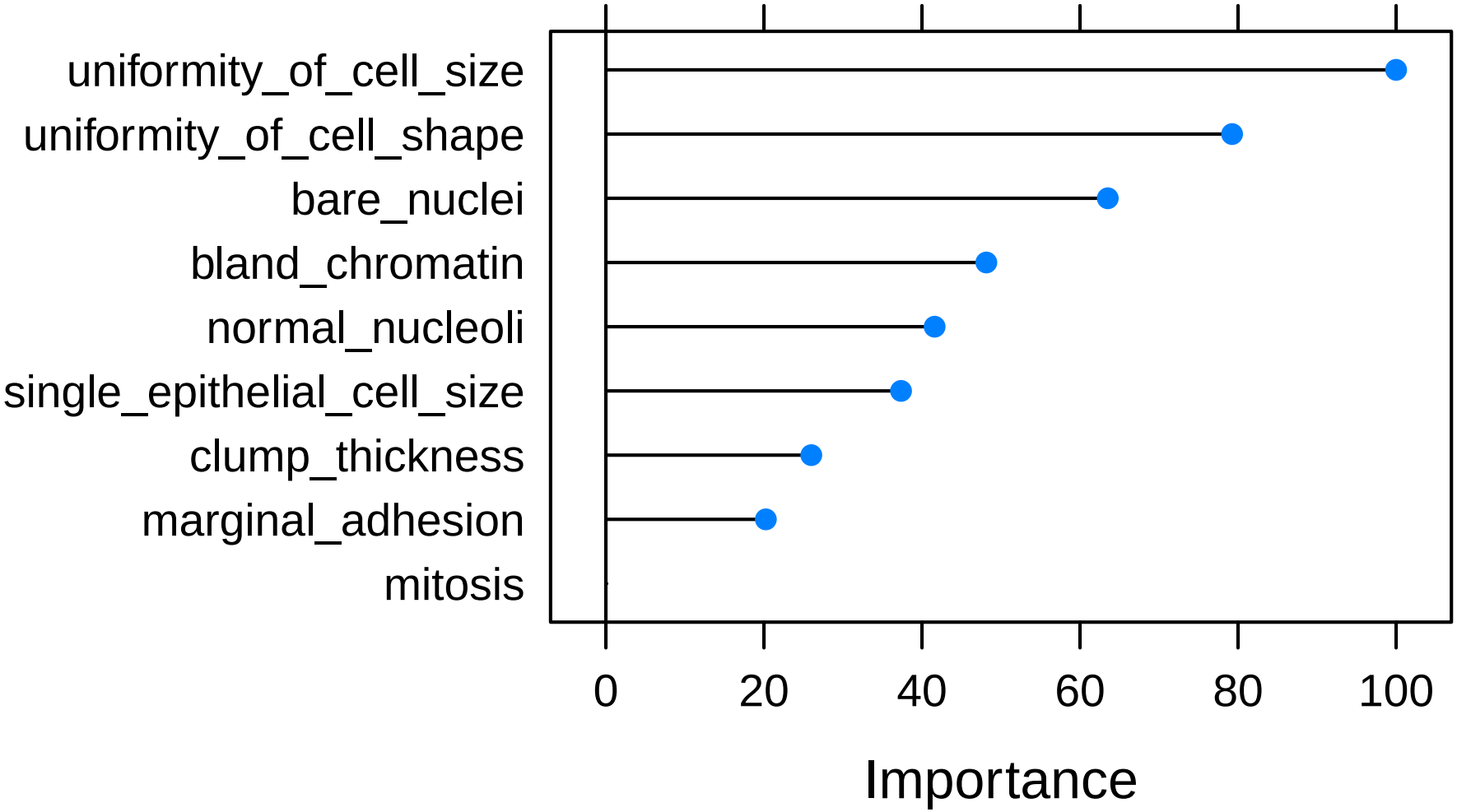
Variáveis Importantes - Escala 100 -> 0

```
importance
```

```
## rf variable importance
##
##                               Overall
## uniformity_of_cell_size        100.00
## uniformity_of_cell_shape        79.25
## bare_nuclei                     63.52
## bland_chromatin                 48.15
## normal_nucleoli                 41.60
## single_epithelial_cell_size     37.35
## clump_thickness                 25.99
## marginal_adhesion               20.24
## mitosis                          0.00
```

Variáveis Importantes – Gráfico

```
plot(importance)
```



Previsões do Modelo de RF de caret

```
predrfx <- predict(model_rf, test_data)
confusionMatrix(predrfx, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction benigno maligno
##    benigno     133       2
##    maligno       4      70
##
##                Accuracy : 0.9713
##                  95% CI : (0.9386, 0.9894)
##     No Information Rate : 0.6555
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9369
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9708
##             Specificity : 0.9722
##          Pos Pred Value : 0.9852
##          Neg Pred Value : 0.9459
##              Prevalence : 0.6555
##          Detection Rate : 0.6364
##    Detection Prevalence : 0.6459
##       Balanced Accuracy : 0.9715
##
##        'Positive' Class : benigno
##
```

Previsões no Formato de Probabilidades

- `type = "prob"` de ‘predict() põe os valores em probabilidades
- Deixa você decidir qual seria o limite para diferenciar entre “benign” e “malignant”
  - Até agora, sempre foi 0.5

```
results <- data.frame(actual = test_data$diag, predict(model_rf, test_data, type = "prob"))
results$prediction <- ifelse(results$benign > 0.5, "benign",
                             ifelse(results$malignant > 0.5, "malignant", NA))
kable(head(results, 8))
```

	actual	benigno	maligno	prediction
3	benigno	1.000	0.000	benign
8	benigno	1.000	0.000	benign
9	benigno	0.964	0.036	benign
12	benigno	1.000	0.000	benign
13	maligno	0.560	0.440	benign
17	benigno	1.000	0.000	benign
19	maligno	0.036	0.964	NA
21	maligno	0.190	0.810	NA