432 Class 20 Slides

thomaselove.github.io/432

2022-03-29

Today's Agenda

- Working with weights in regression models
  - Logistic regression on aggregated data
  - Using survey weights from NHANES
- Classification & Regression Trees and the Titanic

Part 1: Working with Weights in Regression

Setup for Part 1

```
library(here); library(magrittr)
library(janitor); library(knitr)
library(rms)
library(broom)

library(survey)
library(nhanesA); library(haven)

library(tidyverse)
theme_set(theme_bw())
```

Part 1A. Logistic regression on aggregated data

Colorectal Cancer Screening Data

The `screening.csv` data (imported into the R tibble `colscr`) are simulated. They mirror a subset of the actual results from Better Health Partnership's original pilot study of colorectal cancer screening in primary care clinics in Northeast Ohio.

```
colscr <- read_csv(here("data/screening.csv")) %>%
    type.convert(as.is = FALSE)
```

Available to us are the following variables

We have 26 rows, one for each `location` (labeled A-Z), which are part of four `systems`, labeled Sys_1 through Sys_4 each of which contain 6 or 7 `locations`.

Variable	Description	Range
`subjects`	# of subjects reported by clinic	803, 7677
`screen_rate`	prop. of `subjects` who were screened	0.64, 0.90
`screened`	# of `subjects` who were screened	572, 6947
`notscreened`	# of `subjects` not screened	231, 1356
`meanage`	mean age of subjects in years	58, 66
`female`	% of subjects who are female	46, 70
`pct_lowins`	% of subjects Medicaid or uninsured	0.3, 51.3

Fitting a Logistic Regression Model to Proportion Data

Here, we have a binary outcome (was the subject screened or not?) but we have aggregated the results to the clinic level.

We can use the counts of the subjects at each clinic (in `subjects`) and the proportion screened (in `screen_rate`) to fit a logistic regression model, as follows:

```
m_screen1 <-  glm(screen_rate ~ meanage + female +
                    pct_lowins + system, family = binomial,
                  weights = subjects, data = colscr)
```

- Note the use of the subject counts as weights to apply an appropriate count to each proportion.

Model m_screeen1

```
tidy(m_screen1, exponentiate = TRUE, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high) %>%
  kable(digits = 3)
```

term	estimate	conf.low	conf.high
(Intercept)	0.265	0.089	0.782
meanage	1.070	1.052	1.089
female	0.981	0.978	0.984
pct_lowins	0.987	0.985	0.988
systemSys_2	0.871	0.830	0.914
systemSys_3	0.961	0.914	1.010
systemSys_4	1.023	0.966	1.084

Fitting Counts of Successes and Failures

Alternatively, we can use the counts of successes and failures within each location to fit our model.

```
m_screen2 <-  glm(cbind(screened, notscreened) ~
                        meanage + female + pct_lowins + system,
              family = binomial, data = colscr)
```

- Now, we don't need weights, since the sample sizes are incorporated into `cbind(screened, notscreened)`.

Model m_screen2

```
tidy(m_screen2, exponentiate = TRUE, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high) %>%
  kable(digits = 3)
```

term	estimate	conf.low	conf.high
(Intercept)	0.265	0.089	0.782
meanage	1.070	1.052	1.089
female	0.981	0.978	0.984
pct_lowins	0.987	0.985	0.988
systemSys_2	0.871	0.830	0.914
systemSys_3	0.961	0.914	1.010
systemSys_4	1.023	0.966	1.084

- Results are, of course, identical to m_screen1.

How does one address this problem in rms?

We can use Glm, which allows for a broader group of generalized linear models than just lrm...

```
d <- datadist(colscr)
options(datadist = "d")

m_screen3 <-  Glm(screen_rate ~ meanage + female +
                    pct_lowins + system,
                  family = binomial, weights = subjects,
                  data = colscr, x = T, y = T)
```

m_screen3

```
exp(m_screen3$coefficients)
```

```
   Intercept      meanage       female   pct_lowins
   0.2652615    1.0703509    0.9808711    0.9866354
system=Sys_2 system=Sys_3 system=Sys_4
   0.8709080    0.9607731    1.0231922
```

- Again, this is the same result that we saw in the other models for the same data.
- Note that `Glm` doesn't provide the `validate` function for this weighted model, so it's not as strong as in other settings.

Part 1B. Incorporating survey weights in a regression model

What are survey weights?

In many surveys, each sampled subject is assigned a weight that is equivalent to the reciprocal of his/her probability of selection into the sample.

$$\text{Sample Subject's Weight} = \frac{1}{Prob(selection)}$$

but more sophisticated sampling designs require more complex weighting schemes. Usually these are published as part of the survey data.

There are several packages available to help incorporate survey weights in R, but I will describe the `survey` package.

An NHANES Example

Let's use the NHANES 2013-14 data and pull in both the demographics (DEMO_H) and total cholesterol (TCHOL_H) databases.

```
demo_raw <- nhanes('DEMO_H')
```

```
Processing SAS dataset DEMO_H     ..
```

```
tchol_raw <- nhanes('TCHOL_H')
```

```
Processing SAS dataset TCHOL_H    ..
```

Detailed descriptions available at

- https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/DEMO_H.htm
- https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/TCHOL_H.htm

Weighting in NHANES

Weights are created in NHANES to account for the complex survey design. A sample weight is assigned to each sample person. It is a measure of the number of people in the population represented by that sample person.

The sample weight is created in three steps:

1. the base weight is computed, which accounts for the unequal probabilities of selection given that some demographic groups were over-sampled;
2. adjustments are made for non-response; and
3. post-stratification adjustments are made to match estimates of the U.S. civilian non-institutionalized population available from the Census Bureau.

Source: https://wwwn.cdc.gov/nchs/nhanes/tutorials/Module3.aspx

Weights in our NHANES data

The `DEMO` file contains two kinds of sampling weights:

- the interview weight (`WTINT2yr`), and
- the MEC exam weight (`WTMEC2yr`)

NHANES also provides several weights for subsamples. A good rule for NHANES is to identify the variable of interest that was collected on the smallest number of respondents. The sample weight that applies to that variable is the appropriate one to use in your analysis.

In our case, we will use the weights from the MEC exam.

What Variables Do We Need?

- SEQN = subject identifying code
- RIAGENDR = sex (1 = M, 2 = F)
- RIDAGEYR = age (in years at screening, topcode at 80)
- DMQMILIZ = served active duty in US Armed Forces (1 = yes, 2 = no)
- RIDSTATR = 2 if subject took both interview and MEC exam
- WTMEC2YR - Full sample 2 year MEC exam weight
- LBXTC = Total Cholesterol (mg/dl)

The first five of these came from the DEMO_H file, and the first and last comes from TCHOL_H.

Merge the DEMO and TCHOL files

```
dim(demo_raw)
```

```
[1] 10175    47
```

```
dim(tchol_raw)
```

```
[1] 8291    3
```

```
joined_df <- inner_join(demo_raw, tchol_raw, by = c("SEQN"))

dim(joined_df)
```

```
[1] 8291   49
```

Create a small analytic tibble

```
nh1314 <- joined_df %>% # has n = 8291
    tibble() %>%
    zap_label() %>% # still have n = 8291
    filter(complete.cases(LBXTC)) %>% # now n = 7624
    filter(RIDSTATR == 2) %>% # still 7624
    filter(RIDAGEYR > 19 & RIDAGEYR < 40) %>% # now n = 1802
    filter(DMQMILIZ < 3) %>% # drop 7 = refused, n = 1801
    mutate(FEMALE = RIAGENDR - 1,
           AGE = RIDAGEYR,
           US_MIL = ifelse(DMQMILIZ == 1, 1, 0),
           WT_EX = WTMEC2YR,
           TOTCHOL = LBXTC) %>%
    select(SEQN, FEMALE, AGE, TOTCHOL, US_MIL, WT_EX)
```

Our nh1314 analytic sample (1/2)

```
Hmisc::describe(nh1314 %>% select(-SEQN))
```

```
> Hmisc::describe(nh1314 %>% select(-SEQN))
nh1314 %>% select(-SEQN)

 5  Variables      1801  Observations
--------------------------------------------------------------------------------
FEMALE
       n  missing distinct     Info      Sum     Mean      Gmd
    1801        0        2    0.749      927   0.5147   0.4998

--------------------------------------------------------------------------------
AGE
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25
    1801        0       20    0.997    29.47    6.695       20       21       24
     .50      .75      .90      .95
      30       34       38       38

lowest : 20 21 22 23 24, highest: 35 36 37 38 39

Value         20    21    22    23    24    25    26    27    28    29    30    31
Frequency     97    88    94   100    87    89    86    73    84    82   106    91
Proportion 0.054 0.049 0.052 0.056 0.048 0.049 0.048 0.041 0.047 0.046 0.059 0.051

Value         32    33    34    35    36    37    38    39
Frequency     95    97    92    73   100    79   103    85
Proportion 0.053 0.054 0.051 0.041 0.056 0.044 0.057 0.047
```

Our nh1314 analytic sample (2/2)

```
-------------------------------------------------------------------------------
TOTCHOL
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25
    1801        0      200        1      181    41.39      127      137      156
     .50      .75      .90      .95
     178      203      230      247

lowest :  69  82  85  87  89, highest: 315 331 343 346 417
-------------------------------------------------------------------------------
US_MIL
       n  missing distinct     Info      Sum     Mean      Gmd
    1801        0        2    0.083       51  0.02832  0.05506

-------------------------------------------------------------------------------
WT_EX
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25
    1801        0     1614        1    44529    27894    17863    19878    24694
     .50      .75      .90      .95
   34642    59561    88228    95152

lowest :   8430.461  10477.935  11042.094  11201.863  11861.047
highest: 123536.360 123848.289 123852.421 124165.148 125680.328
-------------------------------------------------------------------------------
```

Each weight represents the number of people exemplified by that subject.

Create nh_design survey design

```
nh_design <-
    svydesign(
        id = ~ SEQN,
        weights = ~ WT_EX,
        data = nh1314)

nh_design <- update( nh_design, one = 1) # helps with counting
```

Unweighted counts, overall and by sex

```
sum(weights(nh_design, "sampling") != 0)
```

```
[1] 1801
```

```
svyby( ~ one, ~ FEMALE, nh_design, unwtd.count)
```

```
  FEMALE counts se
0      0    874  0
1      1    927  0
```

```
svyby( ~ one, ~ FEMALE + US_MIL, nh_design, unwtd.count)
```

```
    FEMALE US_MIL counts se
0.0      0      0    829  0
1.0      1      0    921  0
0.1      0      1     45  0
1.1      1      1      6  0
```

Weighted counts, overall and by groups

Weighted size of the generalizable population, overall and by groups.

```
svytotal( ~ one, nh_design )
```

```
       total      SE
one 80196108 1104558
```

```
svyby( ~ one, ~ FEMALE * US_MIL, nh_design, svytotal)
```

```
    FEMALE US_MIL        one        se
0.0      0      0 37185326.4 1225990.7
1.0      1      0 40151728.1 1192408.4
0.1      0      1  2509429.8  419477.5
1.1      1      1   349624.1  157476.1
```

Use the survey design to get weighted means

What is the mean of total cholesterol, overall and in groups?

```
svymean( ~ TOTCHOL, nh_design, na.rm = TRUE)
```

```
           mean     SE
TOTCHOL 181.25 1.0172
```

```
svyby(~ TOTCHOL, ~ FEMALE + US_MIL, nh_design,
      svymean, na.rm = TRUE)
```

```
    FEMALE US_MIL  TOTCHOL       se
0.0      0      0 182.3569 1.575994
1.0      1      0 180.0248 1.368408
0.1      0      1 186.6966 5.354835
1.1      1      1 164.1984 6.535223
```

Unweighted Group Means of TOTCHOL

```
nh1314 %>% dplyr::summarize(n = n(), mean(TOTCHOL))
```

```
# A tibble: 1 x 2
      n `mean(TOTCHOL)`
  <int>           <dbl>
1  1801            181.
```

```
nh1314 %>% group_by(FEMALE, US_MIL) %>%
  dplyr::summarize(n = n(), mean(TOTCHOL)) %>%
  kable(digits = 2)
```

FEMALE	US_MIL	n	mean(TOTCHOL)
0	0	829	182.22
0	1	45	187.11
1	0	921	179.71
1	1	6	169.50

Measures of uncertainty (Survey-Weighted)

Mean of total cholesterol within groups with 90% CI?

```
grouped_result <- svyby(~ TOTCHOL, ~ FEMALE + US_MIL,
                        nh_design, svymean, na.rm = TRUE)
coef(grouped_result)
```

```
     0.0      1.0      0.1      1.1
182.3569 180.0248 186.6966 164.1984
```

```
confint(grouped_result, level = 0.90)
```

```
          5 %     95 %
0.0 179.7646 184.9492
1.0 177.7739 182.2756
0.1 177.8887 195.5045
1.1 153.4489 174.9478
```

- Get standard errors with `se(grouped_result)`, too.

Perform a survey-weighted generalized linear model

Actually, we'll run two models, first without and second with an interaction term between FEMALE and US_MIL.

```
glm1_res <- svyglm(
    TOTCHOL ~ AGE + FEMALE + US_MIL,
    nh_design, family = gaussian())
```

```
glm2_res <- svyglm(
    TOTCHOL ~ AGE + FEMALE * US_MIL,
    nh_design, family = gaussian())
```

Gaussian family used to generate linear regressions here.

Model 1 Results

```
summary(glm1_res)
```

```
Call:
svyglm(formula = TOTCHOL ~ AGE + FEMALE + US_MIL, design = nh_
    family = gaussian())

Survey design:
update(nh_design, one = 1)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 137.1293     5.0039  27.404   <2e-16 ***
AGE           1.5647     0.1697   9.222   <2e-16 ***
FEMALE       -3.2123     2.0092  -1.599    0.110
US_MIL        0.5936     5.0392   0.118    0.906
---
Signif. codes:
```

Model 2 Results

```
summary(glm2_res)
```

```
Call:
svyglm(formula = TOTCHOL ~ AGE + FEMALE * US_MIL, design = nh_
    family = gaussian())

Survey design:
update(nh_design, one = 1)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   136.8639     5.0061  27.340  < 2e-16 ***
AGE             1.5676     0.1696   9.244  < 2e-16 ***
FEMALE         -2.8681     2.0285  -1.414  0.15757
US_MIL          3.4267     5.4710   0.626  0.53117
FEMALE:US_MIL -22.0653     8.5522  -2.580  0.00996 **
---
```

Do tidy and glance work?

```
tidy(glm2_res)
```

```
# A tibble: 5 x 5
  term          estimate std.error statistic   p.value
  <chr>            <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)     137.        5.01     27.3   6.87e-138
2 AGE               1.57      0.170     9.24  6.52e- 20
3 FEMALE           -2.87      2.03     -1.41  1.58e-  1
4 US_MIL            3.43      5.47      0.626 5.31e-  1
5 FEMALE:US_MIL   -22.1       8.55     -2.58  9.96e-  3
```

```
glance(glm2_res)
```

```
# A tibble: 1 x 6
  null.deviance df.null   AIC      BIC deviance df.residual
          <dbl>   <int> <dbl>    <dbl>    <dbl>        <dbl>
1      2498023.    1800  22.2 2341671. 2341633.         1796
```

Part 2: CART and the Titanic

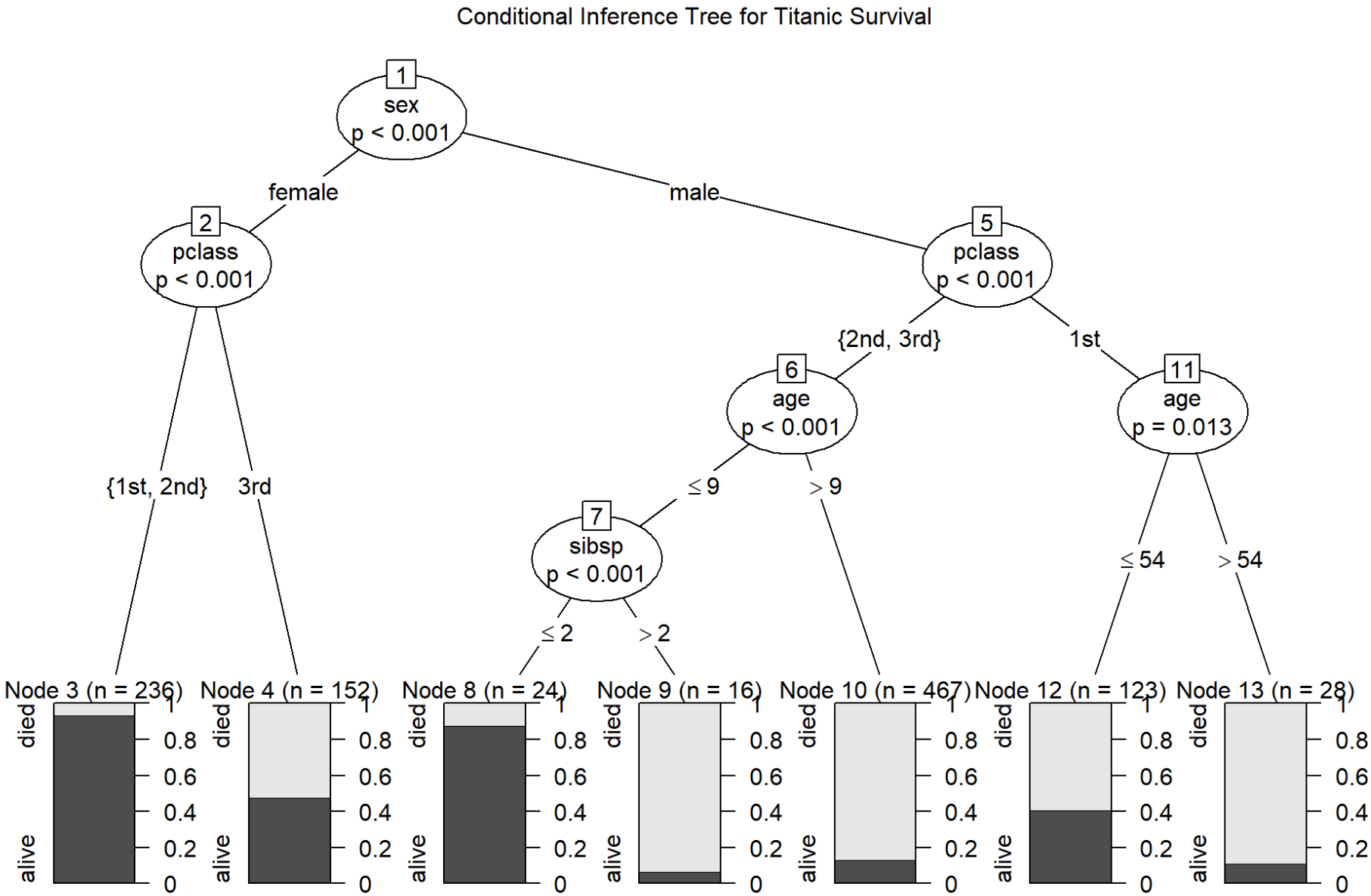
Packages and Setup for our CART work

```
library(readxl)
library(Hmisc)
library(rpart) # new today
library(rpart.plot) # new today
library(party) # new today
library(tidyverse)
```

CART and the Titanic

My goal with this material is to:

1. take a batch of data about survival on the Titanic and
2. build a conditional inference tree to help classify passengers into groups in a way that makes accurate predictions.


Conditional Inference Tree for Titanic Survival
1
sex
p < 0.001
female
male
2
pclass
p < 0.001
5
pclass
p < 0.001
{1st, 2nd}
3rd
{2nd, 3rd}
1st
6
age
p < 0.001
11
age
p = 0.013
≤ 9
> 9
≤ 54
> 54
7
sibsp
p < 0.001
≤ 2
> 2
Node 3 (n = 236)
Node 4 (n = 152)
Node 8 (n = 24)
Node 9 (n = 16)
Node 10 (n = 467)
Node 12 (n = 123)
Node 13 (n = 28)
died
alive
0
0.2
0.4
0.6
0.8
1

Sources and Resources

There are three new libraries we'll use in this work, that you'll have to install, called `rpart`, `rpart.plot` and `party`. The rest of this is (generously) 10% original material from me. Sources:

- statmethods.net (lots of the descriptions are here)
- CART with rpart (uses the titanic data)
- rpart vignette
- party vignette
- milbo.org (tutorial on rpart.plot for tree plotting)

Less immediately useful for this document, but useful in other settings were:

- CART talk
- RandomForests is old, but still useful

The data set

The dataset was compiled by Frank Harrell and Robert Dawson and Philip Hind, among others.

- The `titanic3.xls` data I provide on the course website describes the survival status of individual passengers on the Titanic. The data frame does not contain information for the crew, but it does contain actual and estimated ages for almost 80% of the passengers.
- The data are available at this link at Vanderbilt, and see this file for additional details.

Some initial tidying

```
titan <- read_excel("data/titanic3.xls") %>%
  mutate(pclass = factor(pclass, levels = c(1, 2, 3),
                         labels = c("1st", "2nd", "3rd")),
         sex = factor(sex),
         survived = factor(survived, levels = c(0, 1),
                           labels = c("died", "alive"))) %>%
  select(pclass, survived, sex, age, sibsp, parch, name) %>%
  na.omit
```

The Variables

We're just going to look at the first six variables here, ignoring the passenger's `name`.

- **pclass** = passenger class (1 = 1st, 2 = 2nd, 3 = 3rd), this is a proxy for socio-economic status, with 1 = Upper, 2 = Middle, 3 = Lower
- **survival** = survival (0 = No, 1 = Yes)
- **sex** = male or female
- **age**, in years
- **sibsp**, the number of siblings/spouses aboard
- **parch**, the number of parents/children aboard

Building a Classification Tree

Recursive partitioning is a fundamental tool in data mining. It helps us explore the structure of a set of data, while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome. Paraphrasing the rpart vignette: Classification and regression trees are built by the following process:

1. The single variable is found which best splits the data into two groups (so that the two groups are as "pure" as possible, essentially, is what we mean by "best splits").
2. The data are separated, and then this process is applied separately to each subgroup.
3. and so on recursively until the subgroups either reach a minimum size or until no improvement can be made.

- The tree is trying to make the nodes as decisive as possible, with as few misclassifications as possible.

Step 1 Begin with a small cp value

```
set.seed(432)
tree <- rpart(survived ~ pclass + sex + age + sibsp + parch,
              data = titan,
     control = rpart.control(minsplit = 30, cp = 0.0001))
```

- `minsplit` is the minimum number of observations that must exist in a node in order for a split to be attempted. The default is only 20, but I would like to show you a relatively small tree here.
- cp here stands for complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. You will eventually determine the value for this parameter to use through cross-validation.

Step 2 Pick the tree size that minimizes misclassification rate (prediction error)

```
> printcp(tree)

Classification tree:
rpart(formula = survived ~ pclass + sex + age + sibsp + parch,
    data = titan, control = rpart.control(minsplit = 30, cp = 1e-04))

Variables actually used in tree construction:
[1] age    parch  pclass sex    sibsp

Root node error: 427/1046 = 0.40822

n= 1046

          CP nsplit rel error  xerror     xstd
1 0.4590164      0   1.00000 1.00000 0.037228
2 0.0245902      1   0.54098 0.54098 0.031419
3 0.0187354      3   0.49180 0.51054 0.030764
4 0.0163934      4   0.47307 0.51288 0.030816
5 0.0058548      6   0.44028 0.48478 0.030177
6 0.0046838      8   0.42857 0.46604 0.029729
7 0.0029274     10   0.41920 0.47307 0.029899
8 0.0023419     14   0.40749 0.46838 0.029786
9 0.0001000     17   0.40047 0.48009 0.030066
```

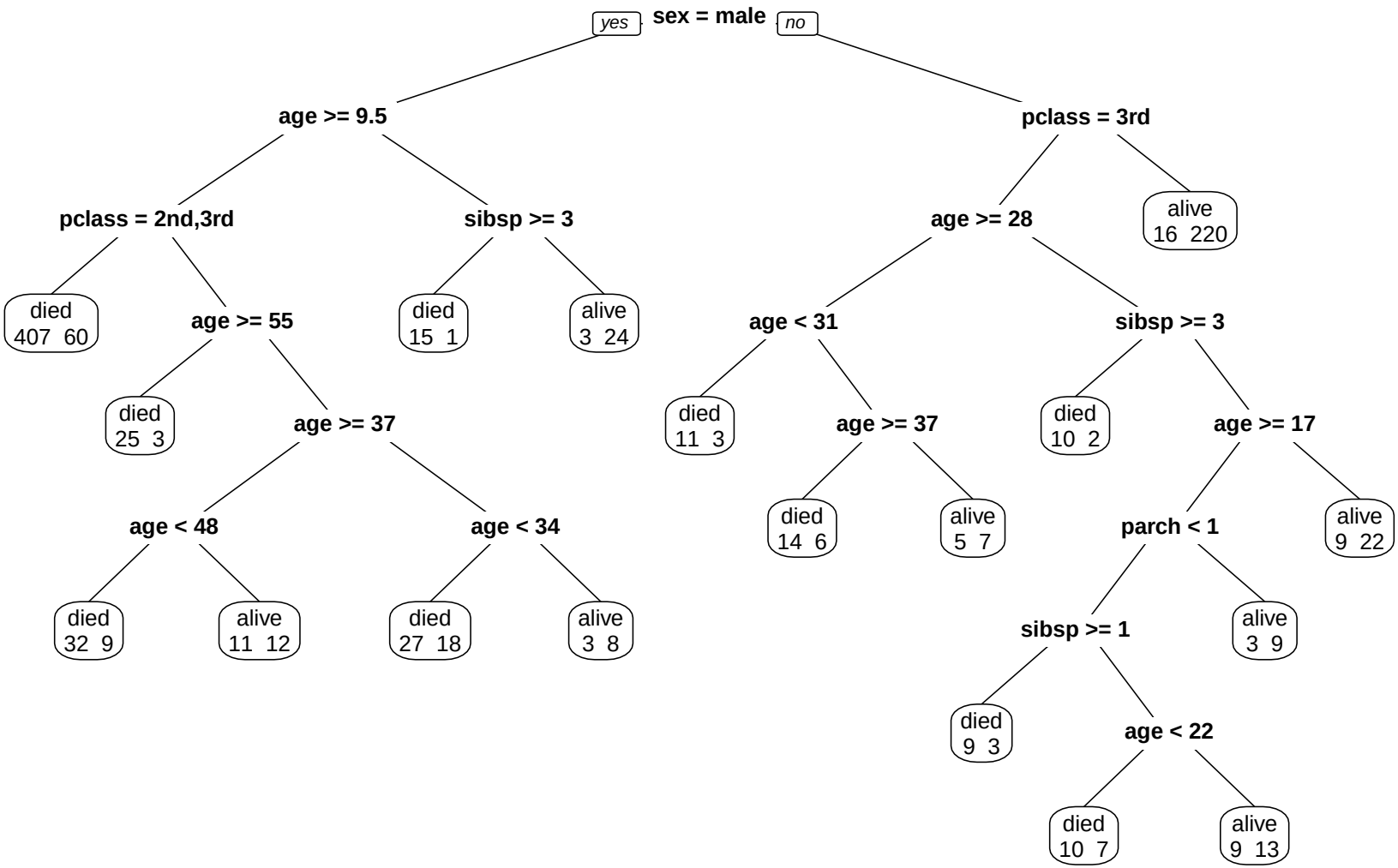
Prune the tree to match cp suggestion

```
bestcp <- tree$cptable[which.min
                       (tree$cptable[,"xerror"]),"CP"]
tree.pruned <- prune(tree, cp = bestcp)
```

Building the Classification Tree Plot

```
prp(tree.pruned, faclen = 0, cex = 0.8, extra = 1,
    main = "Classification Tree for 1,046 Titanic Passengers")
## faclen = 0 means to use full names of the factor labels
## extra = 1 adds number of observations at each node
```

The Classification Tree



A Regression Tree

Suppose we use the same data, but a continuous outcome: age, rather than survival.

```
set.seed(0434)
tree2 <- rpart(age ~ pclass + sex + survived + sibsp + parch,
               data = titan,
               control = rpart.control(minsplit = 20,
                                       cp = 0.0001))
```

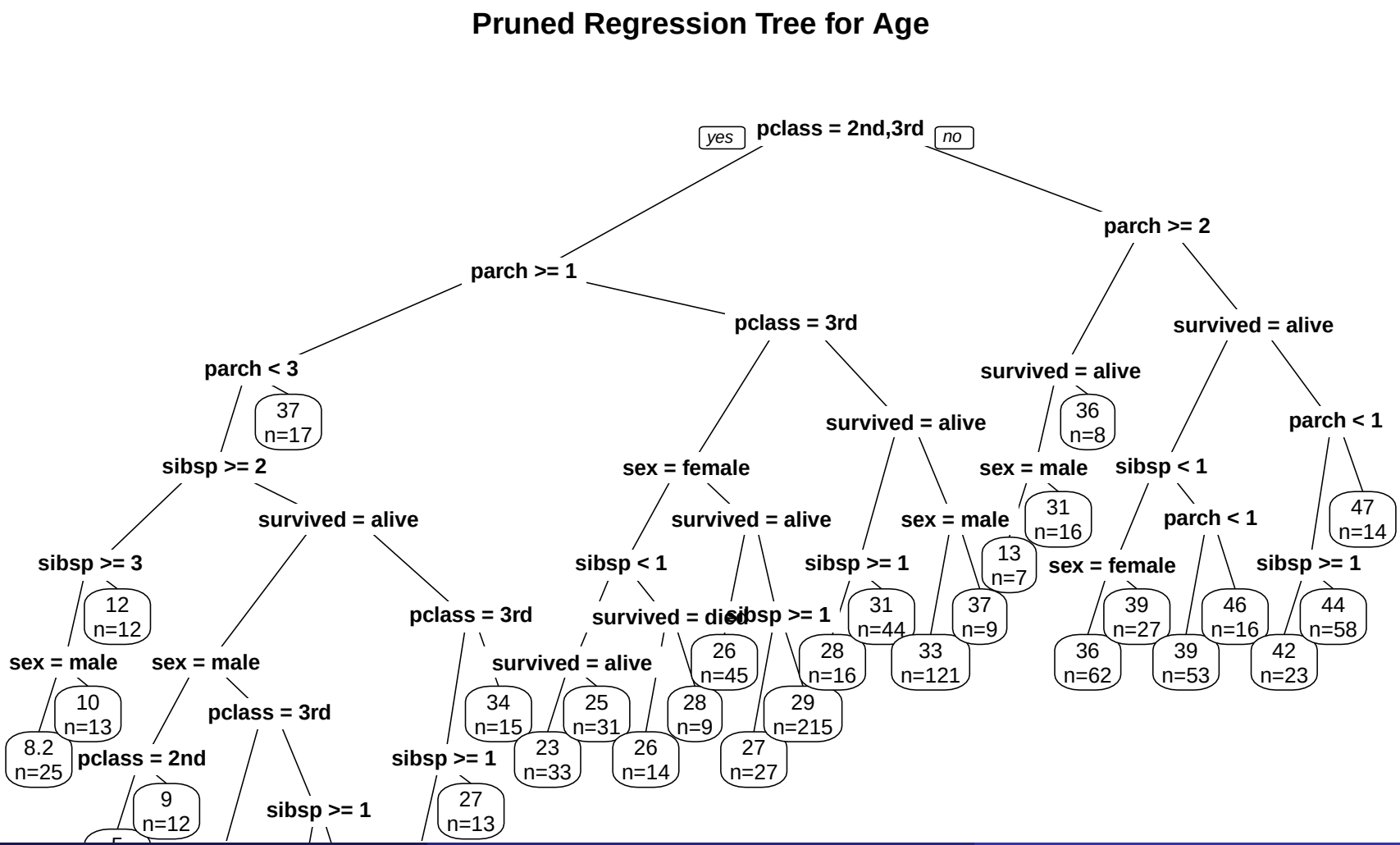
Now identify the best choice of cp, and prune

```
bestcp2 <- tree2$cptable[which.min(
                    tree2$cptable[,"xerror"]),"CP"]

tree2.pruned <- prune(tree2, cp = bestcp)
```

The Regression Tree

```
prp(tree2.pruned, faclen = 0, cex = 0.8, extra = 1,
    main = "Pruned Regression Tree for Age")
```



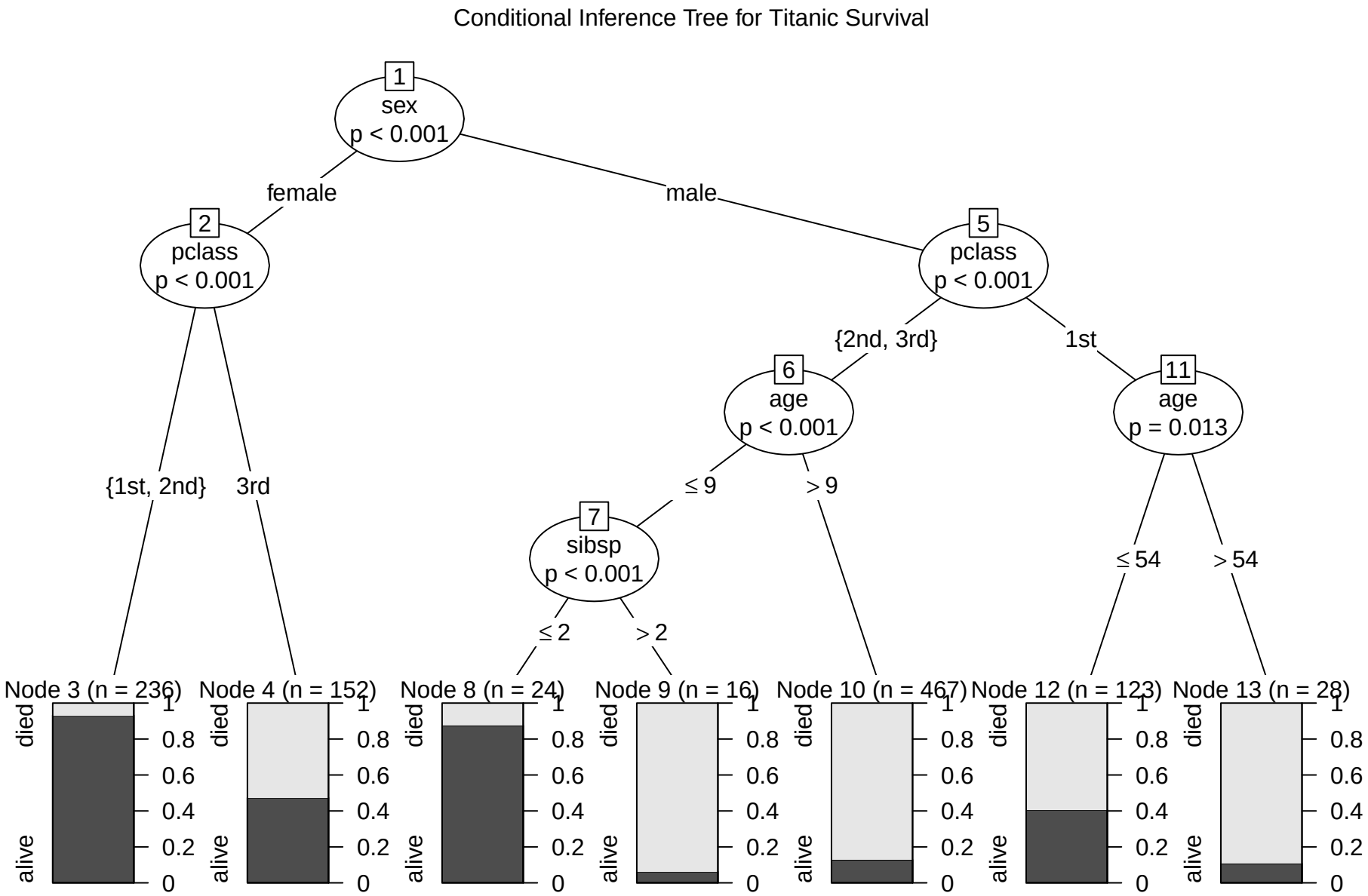
Conditional Inference Trees via party

The party package gives us nonparametric regression trees for all sorts of outcomes. Let's look at our two examples:

Conditional Inference Tree for survived in Titanic data

```
tree.sur <- ctree(survived ~ pclass + sex + age +
                        sibsp + parch,
                  data = titan)
plot(tree.sur,
     main = "Conditional Inference Tree for Titanic Survival")
```

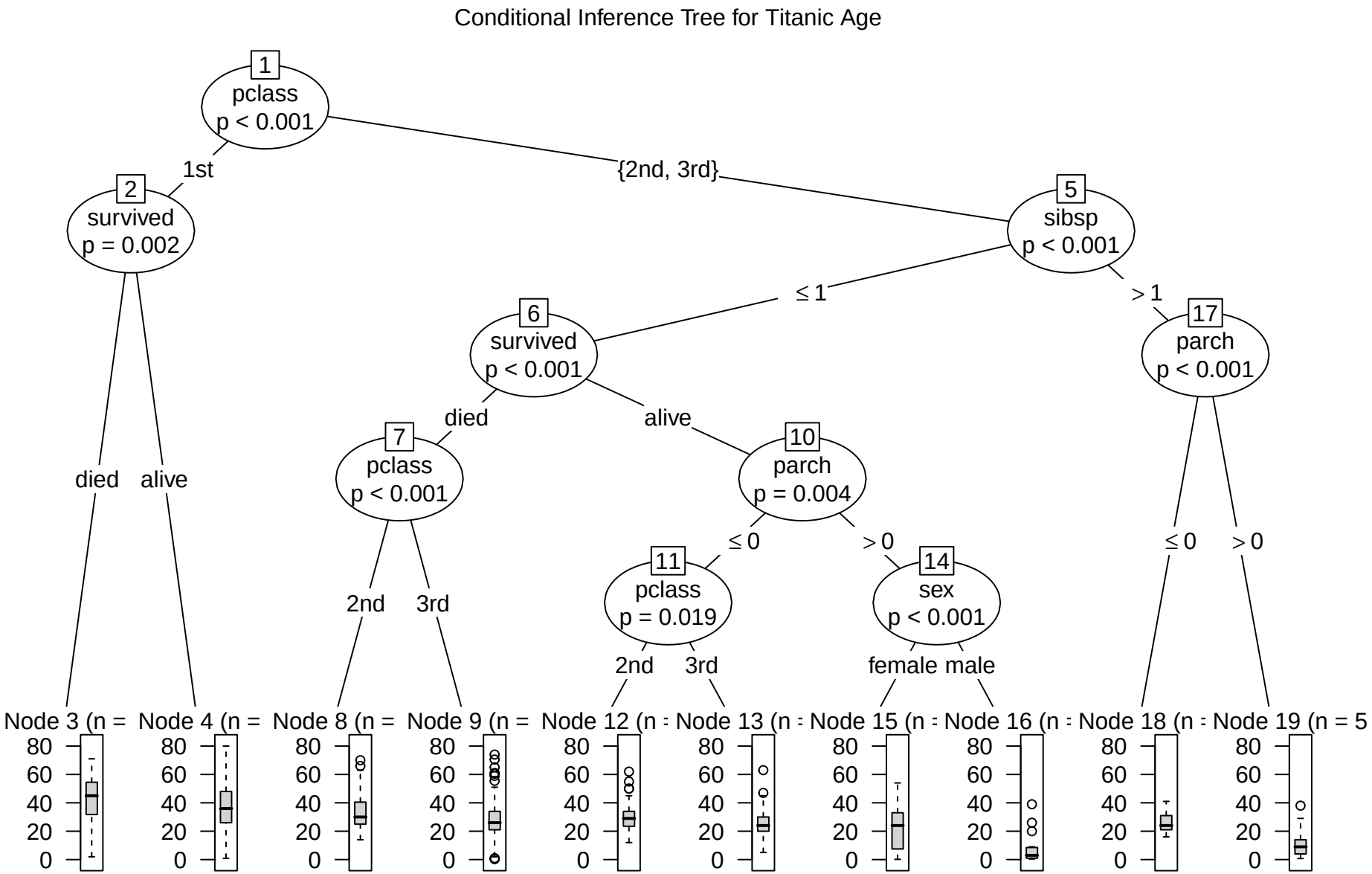
Resulting Tree for `survived`



Conditional Inference Tree for age in Titanic data

```
tree.age <- ctree(age ~ pclass + sex + survived +
                        sibsp + parch,
                  data = titan)
plot(tree.age,
     main = "Conditional Inference Tree for Titanic Age")
```

Resulting Tree for age



Next Time

Ridge Regression and The Lasso