Lab 7 – Classification

This lab accompanies ISLR Chapter 4. Part 1 is a simple logistic regression example. Part 2 uses linear discriminant analysis (LDA) to predict vegetation types across California.

Please create a Markdown file to answer the questions.

Part 1 - Logistic regression

We have done a survey and collected transportation mode choice data. For 16 respondents we know whether they travel to their work by train or not. We have also estimated the travel time to the same destination when going by car.

```
# y: travels by train or not
y <- c(0,1,0,1,0,1,0,1,1,1,0,0,0,0,1,1)
# x: travel time by car
x <- c(32,89,50,49,80,56,40,70,72,76,32,58, 12, 15, 110, 120)
```

We would like to know whether car travel time influences the choice to go by train or not. This is a binary choice, either you go by train or you don't. Logistic regression is the classic approach to this type of problem. We can fit a logistic model with the `glm` function, by specifying that the link function ('family') is 'binomial' — the default value for family is 'gaussian', i.e. ordinary least squares regression.

```
gm <- glm(y~x, family=binomial)
summary(gm)
##
## Call:
## glm(formula = y ~ x, family = binomial)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.96706  -0.48874  -0.04451   0.67740   1.54105
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.93669    2.52358  -1.956   0.0504 .
## x            0.08394    0.04159   2.019   0.0435 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22.181  on 15  degrees of freedom
## Residual deviance: 12.691  on 14  degrees of freedom
## AIC: 16.691
##
## Number of Fisher Scoring iterations: 5
confint(gm)
## Waiting for profiling to be done...
##                    2.5 %     97.5 %
## (Intercept) -11.68456865 -1.1943668
## x             0.02328082  0.1961966
```

We see that the p-value for 'x' is pretty small given that we only have 16 observations, suggesting that x indeed affects y. There is no R^2 value for a logistic model (although there are some *pseudo* R^2 measures that are similar but not widely used). Instead we get measures of deviance (similar to variance) for the Null model (a model that only has an intercept, that is, the mean value) and for the residuals (difference between null and the actual model used). The smaller the residual deviance, the better the model.

We can see if the amount of residual deviance is significant (low p value), treating it as a Chi-square value.

```
pchisq(gm$deviance, gm$df.residual)
## [1] 0.44903
```

It is not significant at all in this case. That is good, because a low p-value would suggest a lack of fit of the model .

We can also use an analysis of deviance to compare the few model to simpler models. In this case, we have only one predictor variable, so there is not much simplification that can be done, but it does show that it is better to include `x` than to not include it.

```
anova(gm, test="Chisq")
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: y
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                    15     22.181
## x     1   9.4895        14     12.691 0.002066 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Just to show how that works by adding two new variables

```
x2 <- x * x
z <- runif(length(x))
gm2 <- glm(y ~ x + x2 + z, family=binomial)
anova(gm2, test="Chisq")
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: y
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                    15     22.181
## x     1   9.4895        14     12.691 0.002066 **
## x2    1   0.2858        13     12.405 0.592948
## z     1   1.2844        12     11.121 0.257087
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The table above shows that adding new variables `x2` and random variable`z` as predictors did not do the model any good, and that we should remove them from the model.

Other approaches to assess the quality of the classification are based on a confusion matrix. These are more relevant because you can make one for both model training and for model testing data, whereas the analysis shown above can only be done for the model training data, and thus may be a bit inflated.

If we the `predict` function with this model, we get odd results:

```
round(predict(gm), 1)
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## -2.3  2.5 -0.7 -0.8  1.8 -0.2 -1.6  0.9  1.1  1.4 -2.3 -0.1 -3.9 -3.7  4.3
##   16
##  5.1
```

Shouldn't the values be probabilities (between zero and one)? Yes, but what we got are the logits. We need to use the argument `type=response` to get values that are (back-) transformed from logits to probabilities.

```
round(predict(gm, type='response'), 1)
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
## 0.1 0.9 0.3 0.3 0.9 0.4 0.2 0.7 0.8 0.8 0.1 0.5 0.0 0.0 1.0 1.0
```

Now we can make a confusion matrix

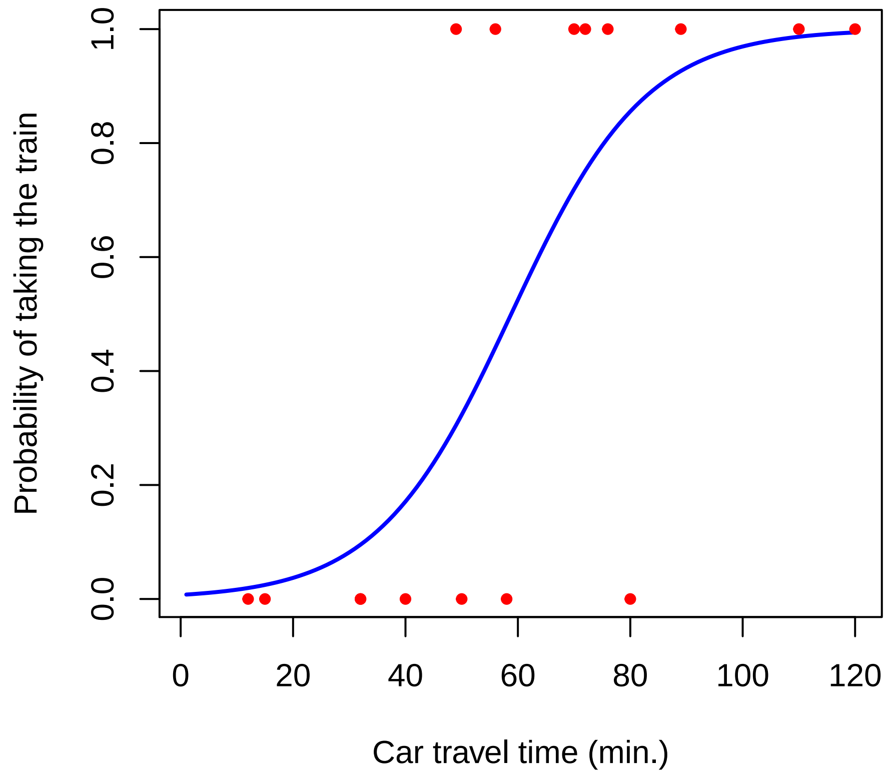
```
ypred <- round(predict(gm, type='response'))
ypred
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
##  0  1  0  0  1  0  0  1  1  1  0  0  0  0  1  1
conf <- table(yobs=y, ypred)
conf
##     ypred
## yobs 0 1
##    0 7 1
##    1 2 6
```

There are many more cases on the diagonal than on the off-diagonal, so the fraction correctly classified is high:

```
sum(diag(conf)) / sum(conf)
## [1] 0.8125
```

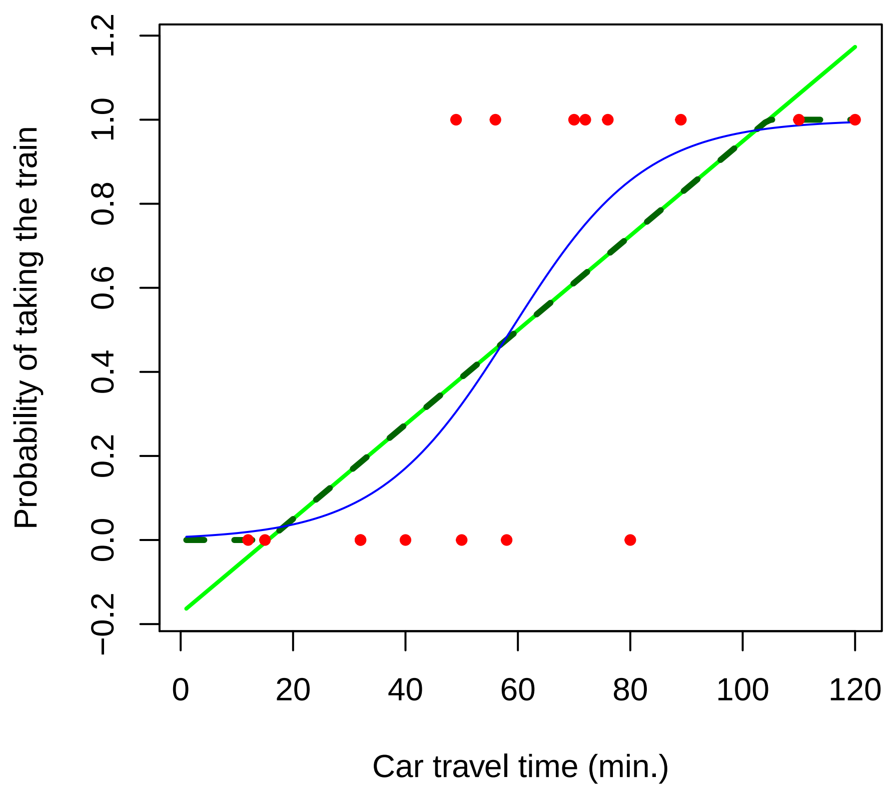
Let's plot the result by predicting mode choice for car-distances between 1 and 120 km.

```
# distances
d <- 1:120
# predictions
pfit <- predict(gm, data.frame(x=d), type='response')
plot(d, pfit, type='l', lwd=2, ylab='Probability of taking the train', xlab='Car travel time (min.)', c
# add the observations
points(x, y, col='red', pch=20)
```



This gives us a nice S-curve. The observed cases are either 0 or 1, but the *probability* of taking the train or not can be any value in between 0 and 1. Let's compare with OLS (Ordinary Least Square) regression. We use `glm` again, but with the (default) Gaussian link function.

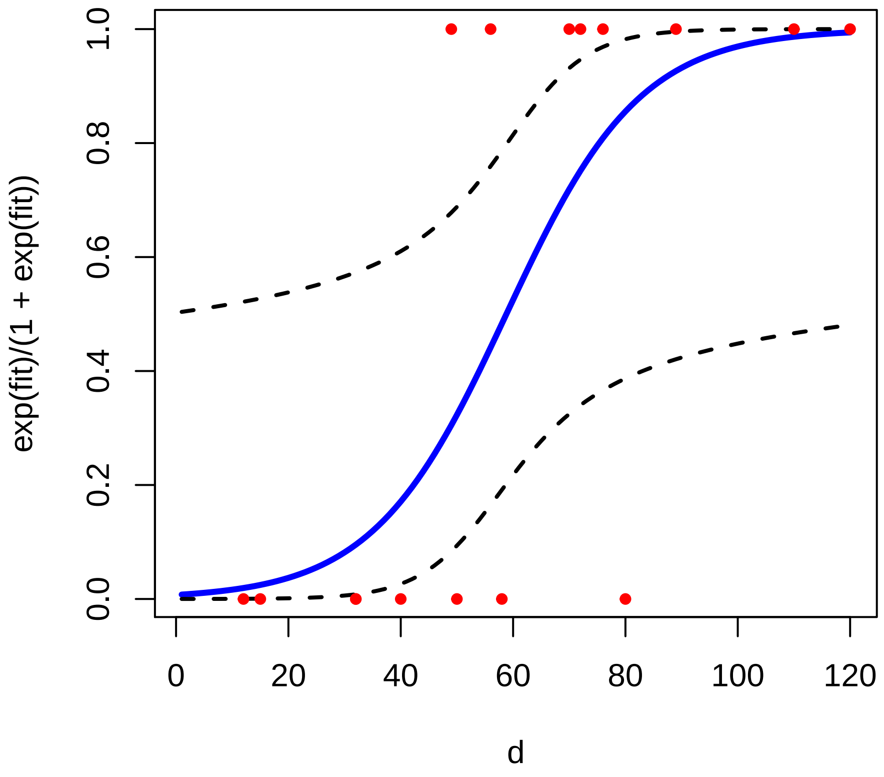
```
ols <- glm(y~x, family='gaussian')
pols <- predict(ols, data.frame(x=d))
plot(d, pols, type='l', lwd=2, ylab='Probability of taking the train', xlab='Car travel time (min.)', co

# a trick to avoid predicting probabilities < 0 and > 1
pols[pols < 0] <- 0
pols[pols > 1] <- 1
lines(pols, lty=2, lwd=3, col='dark green')

# add logistic prediction and observations
lines(d, pfit, type='l', lwd=1, col='blue')
points(x, y, col='red', pch=20)
```



So, predictions with OLS are not that different in this case, but they need to be 'fixed' to avoid impossible probabilities (and the notion of OLS is simply less consistent with the data).

Showing confidence in our predictions is a bit tricky, but important. It shows that we cannot be very certain about our predictions.

```
# do not use 'type=response'
pred <- predict(gm, data.frame(x=d), se.fit=TRUE)
str(pred)
## List of 3
##  $ fit           : Named num [1:120] -4.85 -4.77 -4.68 -4.6 -4.52 ...
##   ..- attr(*, "names")= chr [1:120] "1" "2" "3" "4" ...
##  $ se.fit        : Named num [1:120] 2.48 2.44 2.4 2.36 2.32 ...
##   ..- attr(*, "names")= chr [1:120] "1" "2" "3" "4" ...
##  $ residual.scale: num 1
with(pred, plot(d, exp(fit)/(1+exp(fit)), col="blue", type='l', lwd=3))
with(pred, lines(d, exp(fit+1.96*se.fit)/(1+exp(fit+1.96*se.fit)), lty=2, lwd=2))
with(pred, lines(d, exp(fit-1.96*se.fit)/(1+exp(fit-1.96*se.fit)), lty=2, lwd=2))
points(x, y, col='red', pch=20)
```



Travel time was codes as numbers (0 or 1). That made it easy to use the same values in OLS. But for the purpose of logistic regression it could have been better (clearer) to use a factor (that's the *R* term for a categorical variable).

```
# y: travels by train = 'Yes', not by train = 'No'
y <- c('No','Yes','No','Yes','No','Yes','No','Yes','Yes','Yes','No','No','No','No','Yes','Yes')
# x: travel time by car
x <- c(32,89,50,49,80,56,40,70,72,76,32,58, 12, 15, 110, 120)
```

But the below gives an error:

```
gm <- glm(y~x, family=binomial)
```

Question 1: How can you fix that? Please complete the code below

```
y <- c('No','Yes','No','Yes','No','Yes','No','Yes','Yes','Yes','No','No','No','No','Yes','Yes')
x <- c(32,89,50,49,80,56,40,70,72,76,32,58, 12, 15, 110, 120)
y <- _________(y)
gm <- glm(y~x, family=binomial)
```

Part 2 - Linear discriminant analysis with spatial data

We have sampled land cover in California. We want to predict land cover everywhere in the state. To do so, we fit a lda model of land cover class as a function of annual average temperature and annual precipitation.

The samples are in the file 'samples.csv'. It has three columns: 'x' and 'y' representing location in the Teale

Albers CRS, and 'whr', a land cover classification.

Please complete the *R* code below to do that and answer the questions by writing *R* code.

```
# read the file with observations
s <- _____('samples.csv', stringsAsFactors=FALSE)
head(s)
```

Question 2. How many records do we have?

As always, answer in *R* code!

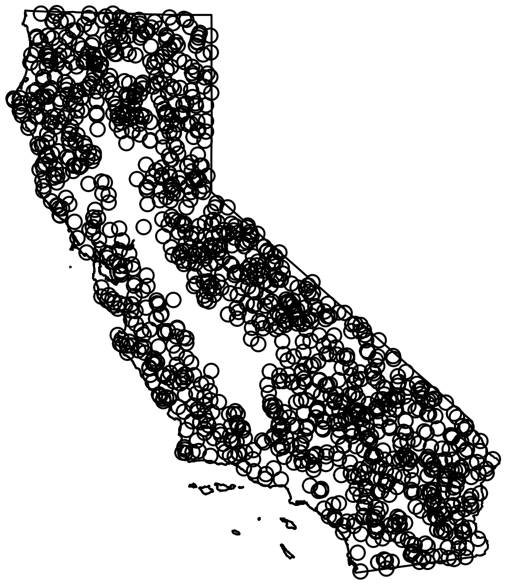
Question 3. Remove the human dominated land cover (urban & agriculture), and also water and wetland from 's' (because you would need additional predictors to adequately model these).

```
s <- subset(s,   )
```

Question 4. How many records do we have now?

To make a map of the locations of the records on top of a map of California, we need to get the boundary of CA, and tranform it to Teale Albers.

Question 5. Finish the code below

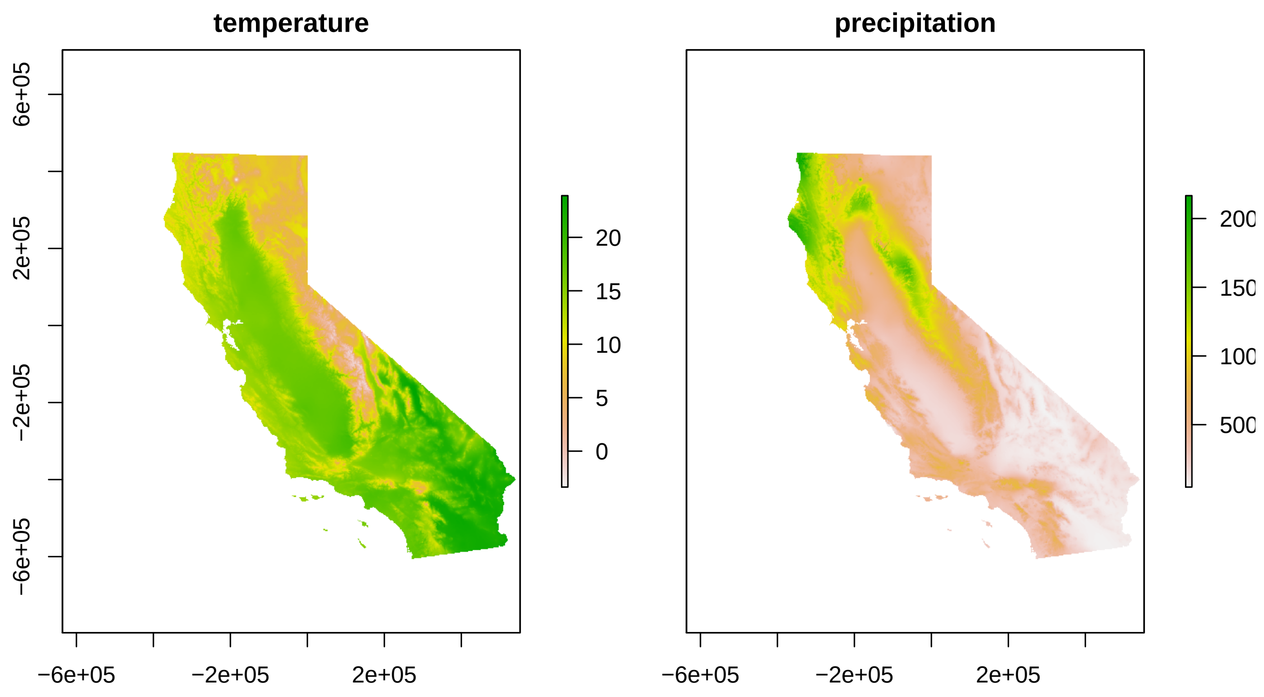


```
library(raster)

usa <- getData('GADM', country='USA', level=1)
```

```
ca <- usa[usa$NAME_1 == 'California', ]
teale_albers <- CRS('+proj=aea +lat_1=34 +lat_2=40.5 +lat_0=0 +lon_0=-120
          +x_0=0 +y_0=-4000000 +datum=NAD83 +units=m')

library(rgdal)
ca <- spTransform(ca, _________)

plot(ca)
points(___________)
```

We did not collect climate data. But we have the locations of our samples, so we can extract that from other data sources. In this case we'll use the the file 'climate.grd' (extracted from the WorldClim database). This file has multiple layers. In that case we can use the 'brick' function to create a multi-layered RasterBrick object.

```
b <- brick('climate.grd')
b
## class       : RasterBrick
## dimensions  : 1270, 1321, 1677670, 2  (nrow, ncol, ncell, nlayers)
## resolution  : 900, 900  (x, y)
## extent      : -636270, 552630, -612690, 530310  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=aea +lat_1=34 +lat_2=40.5 +lat_0=0 +lon_0=-120 +x_0=0 +y_0=-4000000 +datum=NAD83
## data source : D:/gdrive/teaching/200CN/labs/week4/climate.grd
## names       : temperature, precipitation
## min values  :   -5.450492,     37.615426
## max values  :      23.900,      2212.701
plot(b)
```



Question 6. Use the 'extract' function to extract raster cell values for each point

```
e <- extract(b, ______)
s <- cbind(s, e)
```

```
head(s)
##         x       y     whr temperature precipitation
## 1  347880 -443940  Desert   14.035021      365.6130
## 3   61680    7860 Conifer    2.650328      619.3005
## 4 -137220 -187440   Shrub   11.582366      736.1567
## 5  503580 -455640  Desert   21.955955      107.1574
## 6 -255120  206760 Conifer    6.795314     1435.0140
## 8  298380 -502440   Shrub   13.947427      515.5257
```

Set 20% of the data apart to test the model.

```
set.seed(0)
i <- sample(nrow(s), 0.2 * nrow(s))
test <- s[i,]
train <- s[-i,]
```

Question 7. Check that we have taken about 20% of the samples for model training

Now we have the data to fit a LDA

```
library(MASS)
##
## Attaching package: 'MASS'
## The following objects are masked from 'package:raster':
##
##     area, select
lda <- lda(whr ~ temperature + precipitation, data=train)
lda
## Call:
## lda(whr ~ temperature + precipitation, data = train)
##
## Prior probabilities of groups:
## Barren/Other      Conifer       Desert     Hardwood   Herbaceous
##   0.03252033   0.27337398   0.29776423   0.08231707   0.13516260
##        Shrub
##   0.17886179
##
## Group means:
##              temperature precipitation
## Barren/Other    8.265085      594.2668
## Conifer         8.547464      956.2446
## Desert         18.398225      143.6991
## Hardwood       13.233499      862.5000
## Herbaceous     15.084583      526.4344
## Shrub          10.747338      574.8596
##
## Coefficients of linear discriminants:
##                        LD1          LD2
## temperature    0.216185040 -0.209616109
## precipitation -0.002188569 -0.002677092
##
## Proportion of trace:
##    LD1    LD2
## 0.9444 0.0556
```

Question 8. Make a confusion matrix

```
## [1] 0.6463415
```

```
p <- predict(lda)
tab <- table(_____, p$class)
```

```
tab
##
##                Barren/Other Conifer Desert Hardwood Herbaceous Shrub
##   Barren/Other            0      21     11        0          0     0
##   Conifer                 0     213      4        2          1    49
##   Desert                  0       0    279        0          3    11
##   Hardwood                0      30      4       11         28     8
##   Herbaceous              0      11     59        2         49    12
##   Shrub                   0      39     14        2         37    84
```

Question 9. Compute the fraction correctly classified

```
sum(______(tab)) / ______(tab)
```

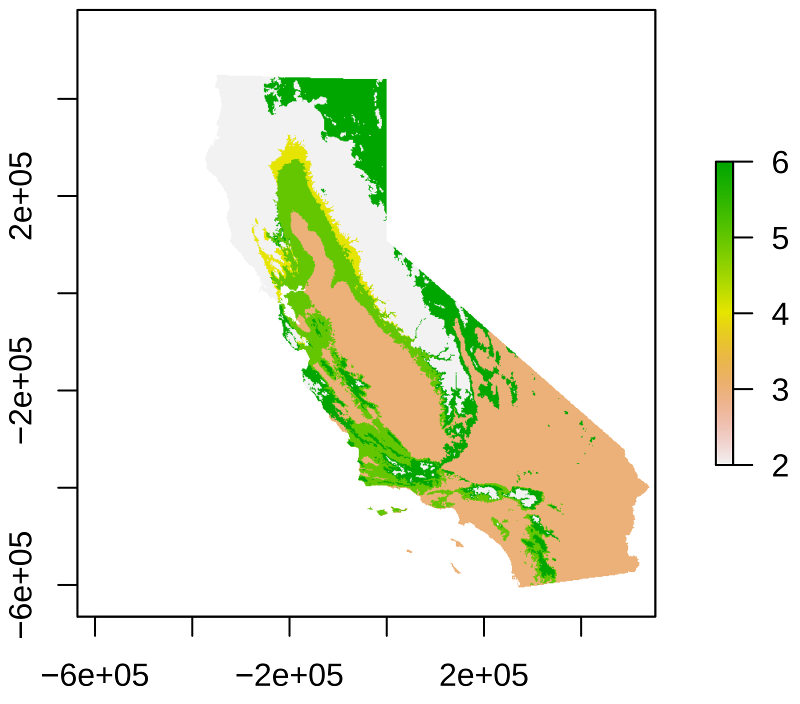
Question 10.Now compute the fraction correctly classified for the model testing data. (I get 0.592)

Question 11. What class seems to be easy to predict well, and what class not?

Question 12. Why might that be?

Now predict to all of California. We use the predict function from the 'raster' package. Note that in this case the first argument is not the fitted model, but a Raster* object that has values for the predictors used to fit the model.

```
pr <- predict(b, lda) # takes a little while
plot(pr)
```



'plot' shows the integer values for the categories, but which value belongs to which category?

```
lda$lev
## [1] "Barren/Other" "Conifer"      "Desert"       "Hardwood"
## [5] "Herbaceous"   "Shrub"
```

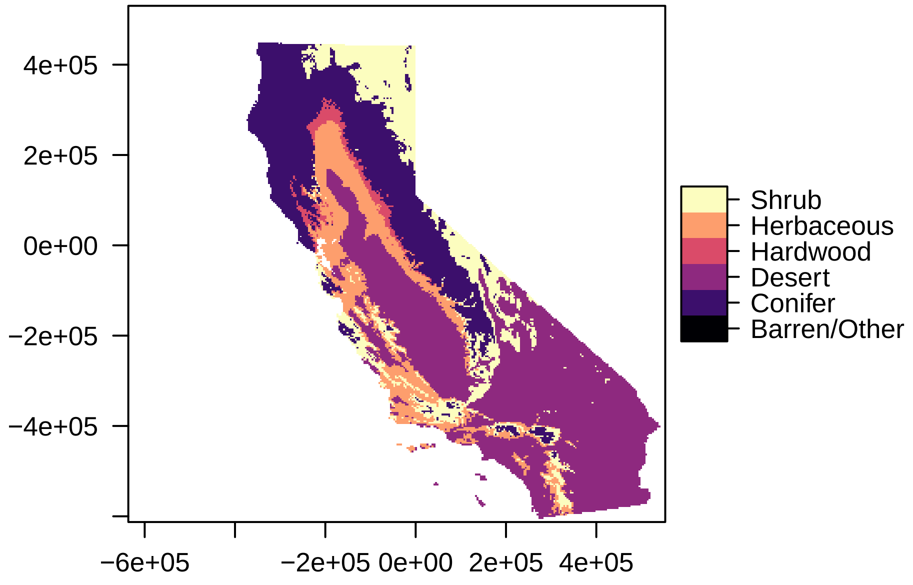
OK, that helps, but that should be an easier way. We can create a 'raster attribute table' that links the cell values (codes) to their description that we can more easily understand. We make a data.frame in which the first column represents the integer value, and the second the class names.

```
levs <- data.frame(ID=1:length(lda$lev), class=lda$lev)
levs
##   ID        class
## 1  1 Barren/Other
## 2  2      Conifer
## 3  3       Desert
## 4  4     Hardwood
## 5  5   Herbaceous
## 6  6        Shrub
levels(pr) <- levs
pr
## class       : RasterLayer
## dimensions  : 1270, 1321, 1677670  (nrow, ncol, ncell)
## resolution  : 900, 900  (x, y)
## extent      : -636270, 552630, -612690, 530310  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=aea +lat_1=34 +lat_2=40.5 +lat_0=0 +lon_0=-120 +x_0=0 +y_0=-4000000 +datum=NAD83
## data source : in memory
## names       : layer
## values      : 2, 6  (min, max)
## attributes  :
##        ID         class
##  from:   1 Barren/Other
##  to  :   6        Shrub
```

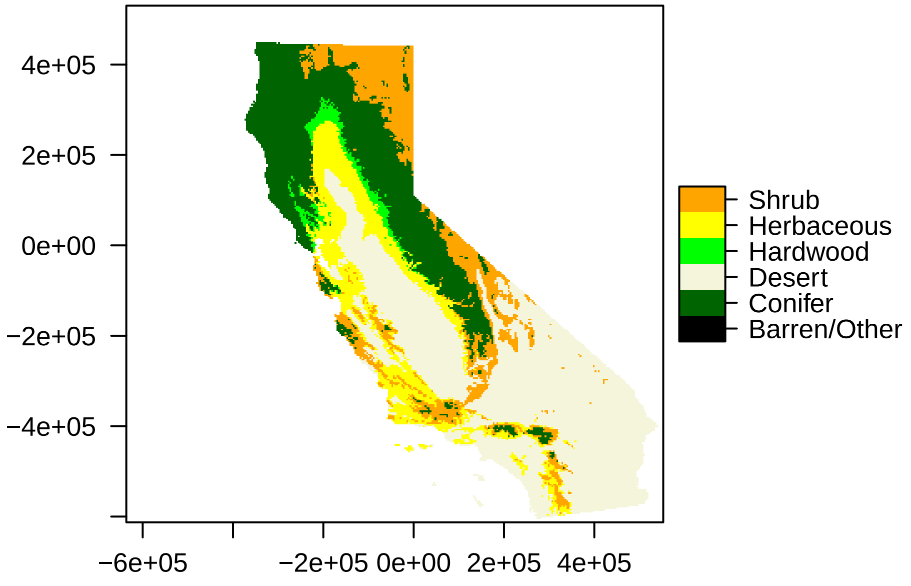
Note that pr now has 'attributes'. This means that the values represent categories (in *R* speak, we say it is a factor). The 'levels' function can be used to set or get them.

Now we can use 'spplot' from raster/sp or 'levelplot' for Raster* objects from the 'rasterVis' package to make a better map.

```r
library(rasterVis)
levelplot(pr)
```



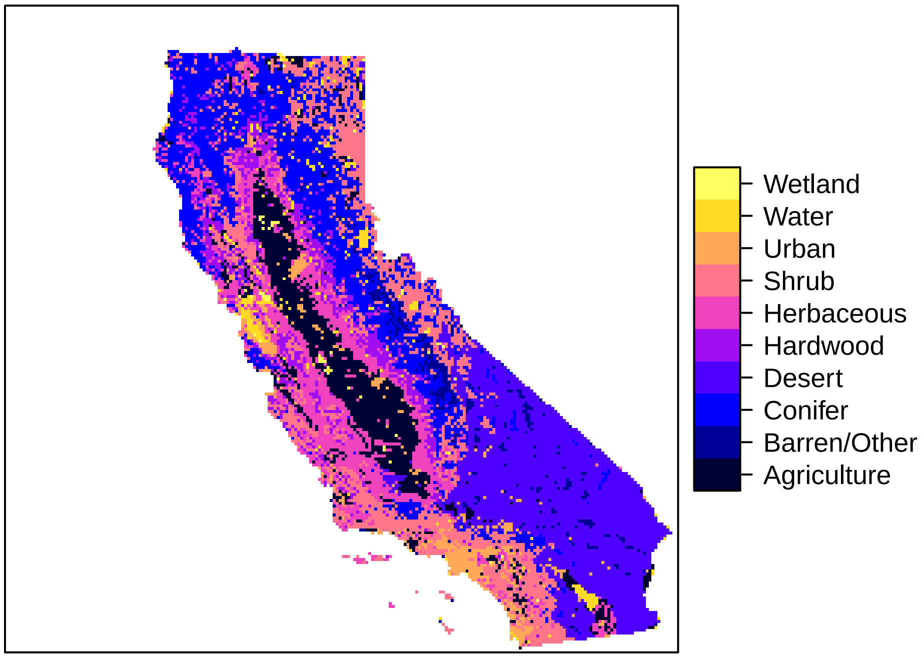
```r
cols <- rev(c('orange', 'yellow', 'green', 'beige', 'dark green', 'black'))
levelplot(pr, col.regions=cols)
```



That is nice. A prediction of the distribution of major vegetation types in California, if there were no urban areas, lakes and wetlands, and other types of disturbance.

Let's compare this to the some real land cover data from FRAP. I have simplified and spatially aggregated the data from 30 m to 900 m.

```
v <- raster('calveg.grd')
levels(v)
## [[1]]
##    ID    WHR10NAME
## 1   1      Conifer
## 2   2        Shrub
## 3   3   Herbaceous
## 4   4      Wetland
## 5   5     Hardwood
## 6   6 Barren/Other
## 7   7        Urban
## 8   8  Agriculture
## 9   9        Water
## 10 10       Desert
spplot(v)
```



Comparing this map with our predictions is difficult. There are additional classes in the 'calveg' data and they are not in the same order. I fix that below (this is a bit advanced). I first merge the two attribute tables.

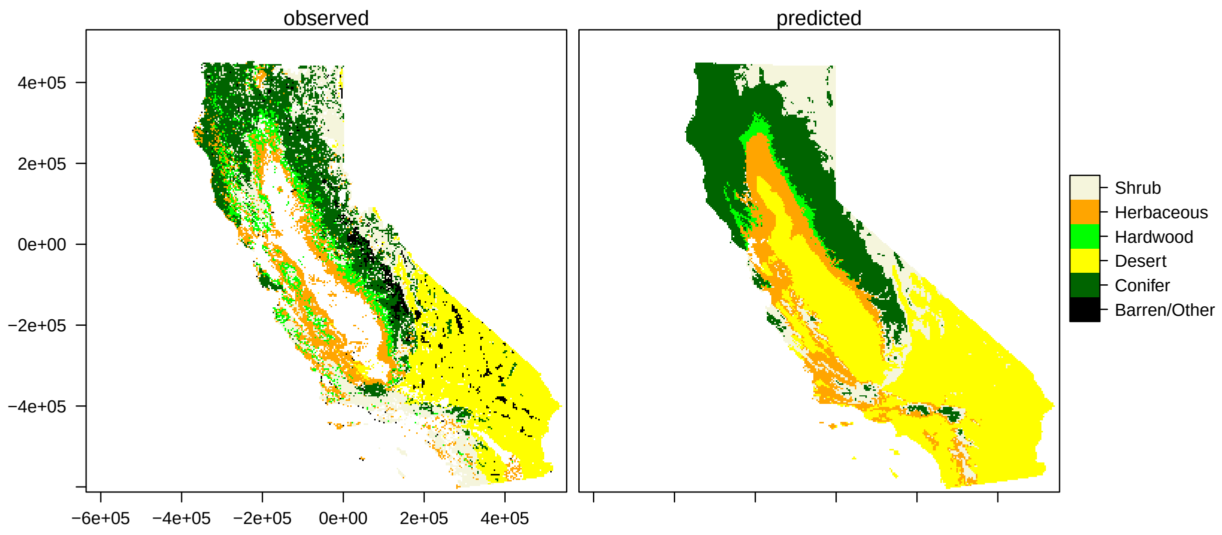
```
m <- merge(levels(v)[[1]], levels(pr)[[1]], by.x='WHR1ONAME', by.y='class', all.x=TRUE)
m
##       WHR1ONAME ID.x ID.y
## 1   Agriculture    8   NA
## 2  Barren/Other    6    1
## 3       Conifer    1    2
## 4        Desert   10    3
## 5      Hardwood    5    4
## 6    Herbaceous    3    5
## 7         Shrub    2    6
## 8         Urban    7   NA
## 9         Water    9   NA
## 10      Wetland    4   NA
```

I then use the second and third columns to reclassify 'v' such that it gets the same coding system as 'pr', and I set that coding system to it.

```
v2 <- reclassify(v, m[,2:3])
levels(v2) <- levels(pr)
```

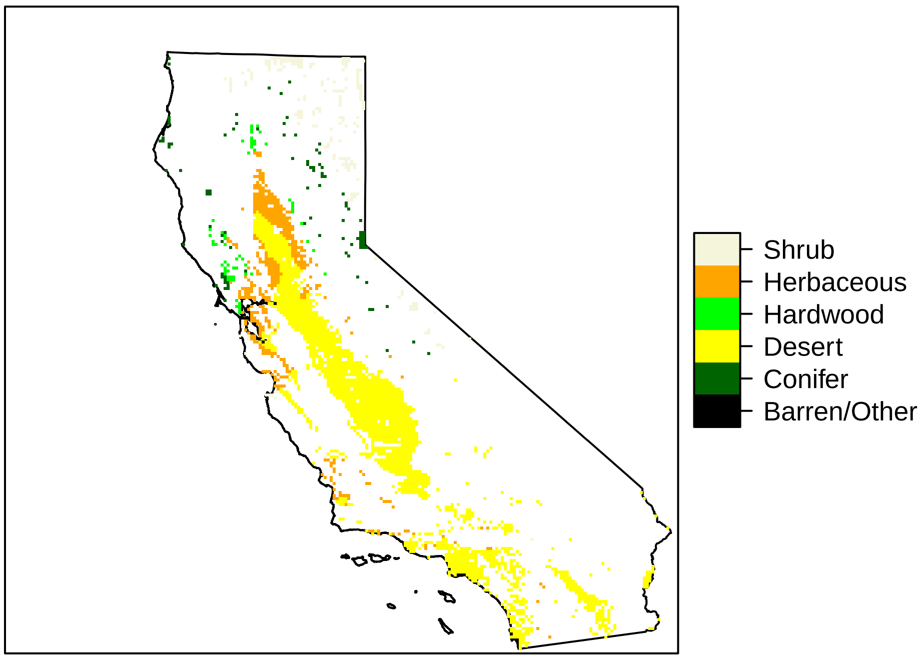
OK, now we just stack the two layers (predicted and observed) and (level)plot them.

```
s <- stack(v2, pr)
names(s) <- c('observed', 'predicted')
cols <- rev(c('beige', 'orange', 'green', 'yellow', 'dark green', 'black'))
levelplot(s, col.regions=cols)
```



Let's look at the predictions of natural vegetation where there is none left. That is, the land cover that our model predicts there would have been in places where we now have agriculture and cities (ignoring lakes and wetlands for now).

```
x <- mask(pr, v2, inverse=TRUE)
levels(x) <- levels(pr)

state <- list("sp.polygons", ca)
spplot(x, col.regions=cols,  sp.layout=state)
```



Question 13. Compare the “undisturbed” vegetation to the actual vegetation. What vegetation type has seen most conversion to agriculture and urban areas (see code below)?

```
f <- freq(x,  useNA='no')
merge(levels(x), f, by=1)
##   ID      class count
## 1  2    Conifer  3401
## 2  3     Desert 62920
## 3  4   Hardwood  1894
## 4  5 Herbaceous 14985
## 5  6      Shrub  6669
```

Question 14. How would you (conceptually or via *R*) test whether this is different from what could be expected by chance?