R Cheat Sheet for CS4375

Getting Started

`getwd()`	returns working dir
`setwd("dir")`	set wd to dir
`dir()`	show files in current dir
`ls()`	list objects in workspace
`rm("x")`	remove x from workspace
`?topic`	get help on topic
`?fun()`	get help on function
`library(x)`	load package x
`install.packages("x")`	install package x

Load Data

`data()`	list R built-in data sets
`data(iris)`	load built-in data set
`df <- read.table("x")`	read text file
`df <- read.csv("x")`	read csv file
`read.csv("x", header=TRUE)`	file has header
`read.csv("x", na.strings="NA")`	interpret NA

Also: download.file("http://...", destfile="x")

Avoid column names with blank spaces.

Common Data Structures

Data Frames

`is.data.frame(iris)`	returns t/f
`df <- as.data.frame(x)`	convert to data frame
`df <- data.frame(...)`	create a data frame

Columns may be of different data types.

Matrix

`m <- matrix(1:10,nrow=2,ncol=5)`	create matrix
`m <- is.matrix(x)`	returns t/f
`m <- as.matrix(x)`	convert to matrix

All columns must be of same data type.

Vector

`v <- 1:10`	create vector
`v <- seq(1,10)`	using seq
`v <- c("a","b","c")`	combine
`v <- rep(c(TRUE,FALSE),3)`	repeat

All items must be of same data type.

Commmon Data Types

`is.numeric(x)`	returns t/f
`as.numeric(x)`	converts
`df$gender <- factor(df$gender)`	make factor

is. and as. work for numeric, integer, character, etc.

Numeric: integer or real

Integer: integer only

Logical: TRUE, FALSE

Character: "1", "many"

Factor: qualitative data

Indexing Data

Indexing Vectors

`v[2]`	2nd element of v
`v[-2]`	all but 2nd element
`v[2:5]`	elements 2-5
`v[c(1:3,5)]`	elements 1,2,3,5
`v[-c(1:3,5)]`	all but el. 1,2,3,5
`v[v> 3 & v<9]`	elements in range
`v[v %in% c(2,4)]`	elements in set

Indexing starts with 1, not 0!

Indexing Matrices

`m[2,3]`	el. at row 2, col 3
`m[2,]`	row 2
`m[,2]`	col 2
`m[,c(1,5)]`	cols 1, 5

Indexing Data Frames

`df$x`	col "x"
`df$x[df$x>5]`	rows where col x $>$ 5

Useful Data Functions

Vectors

`class(v)`	class
`attributes(v)`	attributes
`length(v)`	number of el
`mode(v)`	returns type
`min(v)`	minimum value
`max(v)`	maximum value
`range(v)`	min and max
`mean(v)`	average
`median(v)`	median
`sum(v)`	sum

Data Frames

`dim(df)`	dimensions
`nrow(df)`	num rows
`ncol(df)`	num cols
`names(df)`	col names
`summary(df)`	summary of df
`head(df)`	1st 6 rows
`tail(df, n=2)`	last 2 rows
`df <- na.omit(df)`	omit rows with na

Selecting

`which.max(v)`	index of max
`which.min(v)`	index of min
`which(v==5)`	indices where el is 5
`rev(v)`	reverse
`sort(v)`	sort
`unique(v)`	duplicates removed
`subset(df, select=c(2,4))`	subset cols 2,4
`subset(df, select=-x)`	subset all but x
`subset(df, y==0, select=-x)`	rows where y is 0 . . .

Data Wrangling

`with(data, expr)`	eval expr on data
`by(data, indices, func)`	eval by factor level
`length(which(df$x==1))`	how many xs are 1
`sapply(df, func)`	apply func to cols of df
`lapply(df, func)`	apply func to cols of df
`tapply(v1, v2, func)`	apply func to v1, grouped by v2

Examples:

```
tapply(iris$Sepal.Length, iris$Species, mean)
length(which(iris$Species=="setosa"))
sapply(iris[,1:3], mean) # returns vector
lapply(iris[,1:3], mean) # returns list
```

Check Correlation

`cor(x,y)`	correlation
`pairs(df)`	correlation matrix

Basic Graphs

`hist(v)`	histogram
`plot(x,y)`	plot
`, xlab="x label"`	optional
`, ylab="y label"`	optional
`, main="heading"`	optional
`, pch=19`	point symbol
`, cex=1.5`	point size
`, color="red"`	point color
`par(mfrow=c(1,2))`	1x2 layout
`cdplot(y~x)`	cond. density

Plot creates scatterplots for numeric data and box plots for qualitative data

Example:

```
plot(Petal.Length, Petal.Width, pch=21,
     bg=c("red","green3","blue")
     [unclass(Species)], main="Iris Data")
```

Basic Coding

`if(cond) expr`	if
`if(cond) expr1 else expr2`	if else
`ifelse(test, yes, no)`	if-else
`for(var in seq) expr`	for
`while(cond) expr`	while
`function (args) expr`	function
# comments	

Loops are often avoided by vector operations.

Example: 5 * v # multiply each el by 5

Operand Surprises

`<-`	assignment
`&`	vector and
`|`	vector or
`&&`	logical and
`||`	logical or

Split Train and Test

```
     by row number
train <- df[1:500]
test <- df[501:700]
     randomly
set.seed(1234)
i <- sample(nrow(df), floor(nrow(df)*0.8), replace=F)
train <- df[i,]
test <- df[-i,]
```

Across Algorithms

`df <- na.omit(df)`	omit rows with na
`df$x <- scale(df$x)`	scale
`df$new <- ifelse(df$x>n,1,0)`	real to binary
`I(x^2)`	inhibit
`contrasts(x)`	factor coding
`summary(model1)`	model summary
`anova(model1, model2)`	compare models

Common Metrics

`accuracy`	mean(pred==test$target)
`mse`	mean(residuals$\hat{2}$)

Linear Regression

`lm1 <- lm(y~x, data=mydata)`	returns model
`lm1 <- lm(y~., data=mydata)`	all predictors
`lm1 <- lm(y~x+z, data=mydata)`	predictors x and z
`lm1 <- lm(y~.-x, data=mydata)`	all but x
`lm1 <- lm(y~.x*z, data=mydata)`	interaction
`lm1 <- lm(log(y)~.x+z, data=mydata)`	transformation
`coef(lm1)`	coefficients
`confint(lm1)`	confidence intervals
`summary(lm1)`	model summary
`predict(lm1, newdata)`	predict on new data
`plot(lm1)`	residual plots

Logistic Regression

```
glm1 <- glm(y~x, family=binomial)
probs <- predict(glm1, newdata=test, type='response')
pred <- ifelse(probs>0.5,1,0)
table(pred=pred, actual=df$response)
mean(pred==df$response)
```

Naive Bayes

```
library(e1071)
nb1 <- naiveBayes(df[,-1], df[,1], data=train)
pred <- predict(nb1, newdata=test[,-1], type="class")
pred <- predict(nb1, newdata=test[,-1], type="raw")
```

Clustering: kNN

```
     knn regression
library(caret)
fit <- knnreg(df[,-1], df[,1], k=3)
pred <- predict(fit, test[,-1])
cor(pred, test[,1]
     knn classify
library(class)
pred <- knn(train=x, test=y, cl=x$Labels, k=3)
mean(pred==x$Labels)
```

Clustering: k-Means

```
km.out <- kmeans(data, centers=5, nstart=25)
fit$centers
```

Clustering: Hierarchical

```
d <- dist(x.scaled)
fit <- hclust(d, method="average")
plot(fit, han=-1, cex=.8)
```

SVM

```
library(e1071)
svm1 <- svm(y~., data=train, kernel="linear", cost=10)
svm1 <- svm(y~., data=train,
     kernel="radial", cost=10, gamma=1)
pred <- predict(svm1, newdata=test[,-1], type="class")
pred <- predict(svm1, newdata=test[,-1], type="raw")
```

Decision Tree

```
     regression tree
library(tree)
tree1 <- tree(y~., df, subset=train)
yhat <- predict(tree1, newdata=test)
mean((yhat-df$response)^2)                MSE
plot(tree1)
text(tree1, pretty=0)
cv.t <- cv.tree(tree1)
plot(cv.t$size, cv.t$dev, type='b')
     classification tree
tree2 <- tree(y~., df)
```