

From “An Introduction to R”, §10.9 Generic Functions  
`methods(class="data.frame")` tells us which generic functions work on objects of this class.

If you want to know which objects have a specialized plotting function, use

```
> methods(plot)
```

With some functions these are not visible, we have to use other finder functions.

```
> getAnywhere("stack.data.frame")
> getS3method("plot", "factor")
> require(arm)
> getMethod("display", "lm")
```

```
> Summize <- function(x) {
+   s
+   if (!is.numeric(x))
+     stop("Data must be numeric")
+   boxplot(x, horizontal = TRUE)
+   list(fiveNum = fivenum(x), IQR = IQR(x), mean = mean(x),
+        SD = sd(x))
+ }
```

Extra arguments ...

```
> Summize <- function(x, ...) {
+   x <- as.numeric(x)
+   boxplot(x, ...)
+   list(fiveNum = fivenum(x), IQR = IQR(x), mean = mean(x),
+        SD = sd(x))
+ }
```

- We use lots of functions to process and plot data.
- We can create our own functions. Simple example: create a function to summarize qualitative data.
  - Input: a vector of data (numeric)
  - Output: Mean and StdDev, five-number summary
  - Plot: boxplot of the data.

```
> Summize <- function(x) {
+   boxplot(x, horizontal = TRUE)
+   c(fivenum(x), mean(x), sd(x))
+ }
> Summize(rexp(50))

[1] 0.02153074 0.40497106 0.84954024 1.37733346 3.72547253
[6] 1.02780953 0.85267821
```

Problem: output is not labeled.

```
> test.means <- function(y1, y2 = NULL, paired = FALSE) {
+   n1 <- length(y1)
+   if (is.null(y2)) {
+     tst <- mean(y1)/sd(y1) * sqrt(n1)
+   }
+   else if (paired)
+     tst <- test.means(y1 - y2, NULL)
+   else {
+     n2 <- length(y2)
+     yb1 <- mean(y1)
+     yb2 <- mean(y2)
+     v1 <- var(y1)
+     v2 <- var(y2)
+     sd <- sqrt((n1 - 1) * v1 + (n2 - 1) *
+               v2)/(n1 + n2 - 2)
+     tst <- (yb1 - yb2)/(sd * sqrt(1/n1 + 1/n2))
+   }
+   tst
+ }
```

## WD function prep 1

Build a function to tabulate fish by length class (25 mm groups) and mark. First do it on one year/site/species.

```
> ruby <- read.csv("../data/Ruby-AllFish.csv", head = TRUE)
> ruby.Ghrn.RBT2006 <- subset(ruby, species == "RBT" &
+   site == "Ghorn" & year == 2006 & length >
+   100)
> out1 <- with(ruby.Ghrn.RBT2006, table(cut(length,
+   seq(100, 475, 25)), mark, trip))
> cbind(out1[, , 1], out1[, , 2])
```

```
      0 1 0 1
(100,125] 0 0 0 0
(125,150] 14 0 6 2
(150,175] 23 0 7 10
(175,200] 26 0 8 7
(200,225] 11 0 3 3
(225,250] 19 0 11 11
(250,275] 19 0 13 10
(275,300] 30 0 13 15
(300,325] 23 0 0 14
(325,350] 17 0 4 8
(350,375] 14 0 8 10
```

Stat 505

R Intro, Day 3

## WD function prep 2

Problem: On trip 1 there are no marks, so we waste the 2nd column

```
> ruby.Ghrn.RBT2006$type <- with(ruby.Ghrn.RBT2006,
+   ifelse(trip == 1, "pass1", ifelse(mark ==
+   1, "both", "pass2")))
> with(ruby.Ghrn.RBT2006, table(cut(length, seq(100,
+   475, 25)), type))
```

```
      type
      both pass1 pass2
(100,125] 0 0 0
(125,150] 2 14 6
(150,175] 10 23 7
(175,200] 7 26 8
(200,225] 3 11 3
(225,250] 11 19 11
(250,275] 10 19 13
(275,300] 15 30 13
(300,325] 14 23 0
(325,350] 8 17 4
(350,375] 10 14 8
(375,400] 2 6 2
```

Stat 505

R Intro, Day 3

## WD function prep 3

Fill in the table for 300–325mm fish:

	Pass 2		
Pass 1	Caught	Missed	
Caught			
Missed			
			N

What are the inputs and outputs? What data problems could occur?

Stat 505

R Intro, Day 3

## WD function attempt 1

Inputs: data, year, species, site.

Output: table

```
> fishTable <- function(data, yr, specs, ste, minLength = 100,
+   maxLength = 500, increment = 25) {
+   myData <- subset(data, species == specs &
+   site == ste & year == yr & length > minLength &
+   length < maxLength)
+   type <- with(myData, ifelse(trip == 1, "pass1",
+   ifelse(mark == 1, "both", "pass2")))
+   results <- with(myData, table(cut(length,
+   seq(minLength, maxLength, increment)),
+   type))[, c(2, 3, 1)]
+   results[, 2] <- results[, 3] + results[, 2]
+   results
+ }
```

Stat 505

R Intro, Day 3

```
> head(ruby.Canyon.RBT2006 <- fishTable(ruby, specs = "RBT"  
+   ste = "Can", yr = 2006))
```

```
      type  
      pass1 pass2 both  
(100,125]     5     4     0  
(125,150]    36    22     6  
(150,175]    82    36    13  
(175,200]    64    24     1  
(200,225]    47    11     3  
(225,250]    56    33    10
```