Formulas in R

Randall Pruim

Calvin College

Anatomy of a formula

```
> y ~ x | z
```

Think: `y` depends on `x` (perhaps differently depending on `z`)

In `lattice` plots:

- `y` is on the vertical axis
- `x` is on the horizontal axis
- `z` splits the plot into multiple subplots (called panels)

In `mosaic` package: `lattice`-like functionality is extended to numerical summaries

Formulas also used to describe linear models (regression, ANOVA, . . .)

Anatomy of a function

Functions in R are called with

- function name (capitalization matters) followed by
- parenthesized
- comma separated list of arguments (identified by name and/or order)

```
> functionname(arg1, name2 = arg2, arg3, ...)
```

So learning a function is a matter of learning its name and its arguments. **The syntax is the same for every function.**

Tab completion in RStudio can help you remember the names and order of arguments.

Anatomy of a function

Example functions:

```
> mean(x)  # compute the mean of something
> mean(x, trim = 0.1)  # trimmed mean
> # load data from a csv file
> mydata <- read.csv("mydata.csv", header = TRUE)
```

```
> head(HELPrct, 2)  # first 2 rows of data


  age anysubstatus anysub cesd d1 daysanysub dayslink
1  37            1    yes   49  3        177      225
2  37            1    yes   30 22          2       NA
  drugrisk e2b female  sex g1b homeless i1 i2 id indtot
1        0  NA      0 male yes   housed 13 26  1     39
2        0  NA      0 male yes homeless 56 62  2     43
  linkstatus link   mcs   pcs pss_fr racegrp satreat
1          1  yes 25.11 58.41      0   black      no
2         NA <NA> 26.67 36.04      1   white      no
  sexrisk substance treat
1       4   cocaine   yes
2       7   alcohol   yes
```

Teaching Tip for Functions

When introducing new functions, focus on

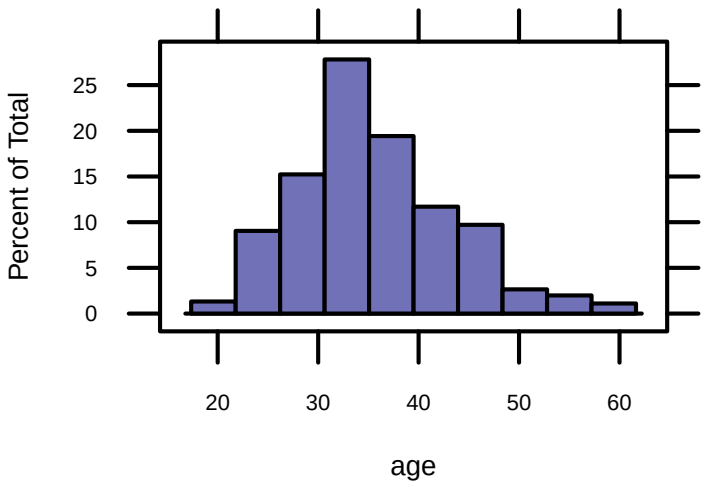
- What the function does (that determines its name)
- What the function needs to know to do its job (that determines the arguments)

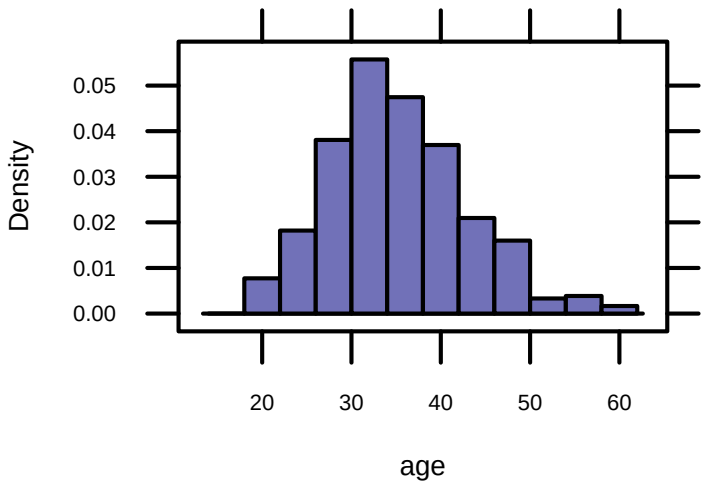
After seeing several examples, you can usually make a good guess about how new functions will work.

1 Quantitative Variable – Histograms

Since we only have one variable, the y and z parts of the formula are omitted.

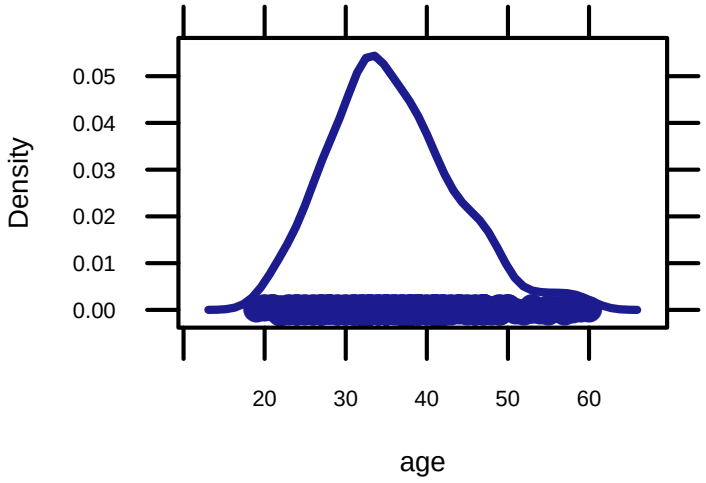
```
> histogram(~age, data = HELPrct)  # lattice
> xhistogram(~age, data = HELPrct, width = 4)  # mosaic
```

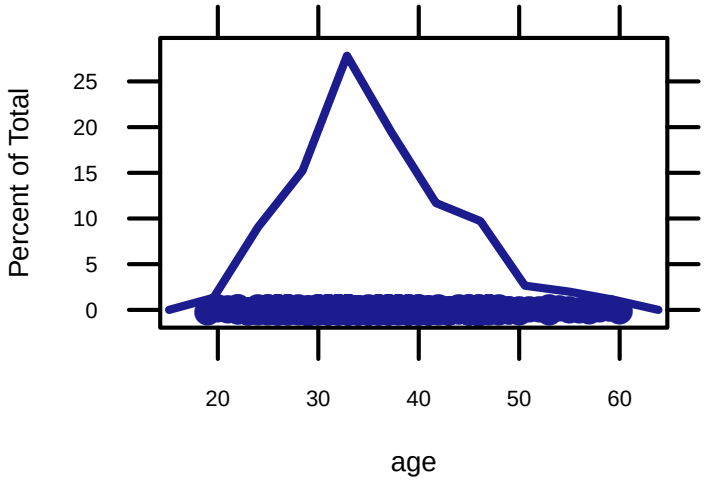




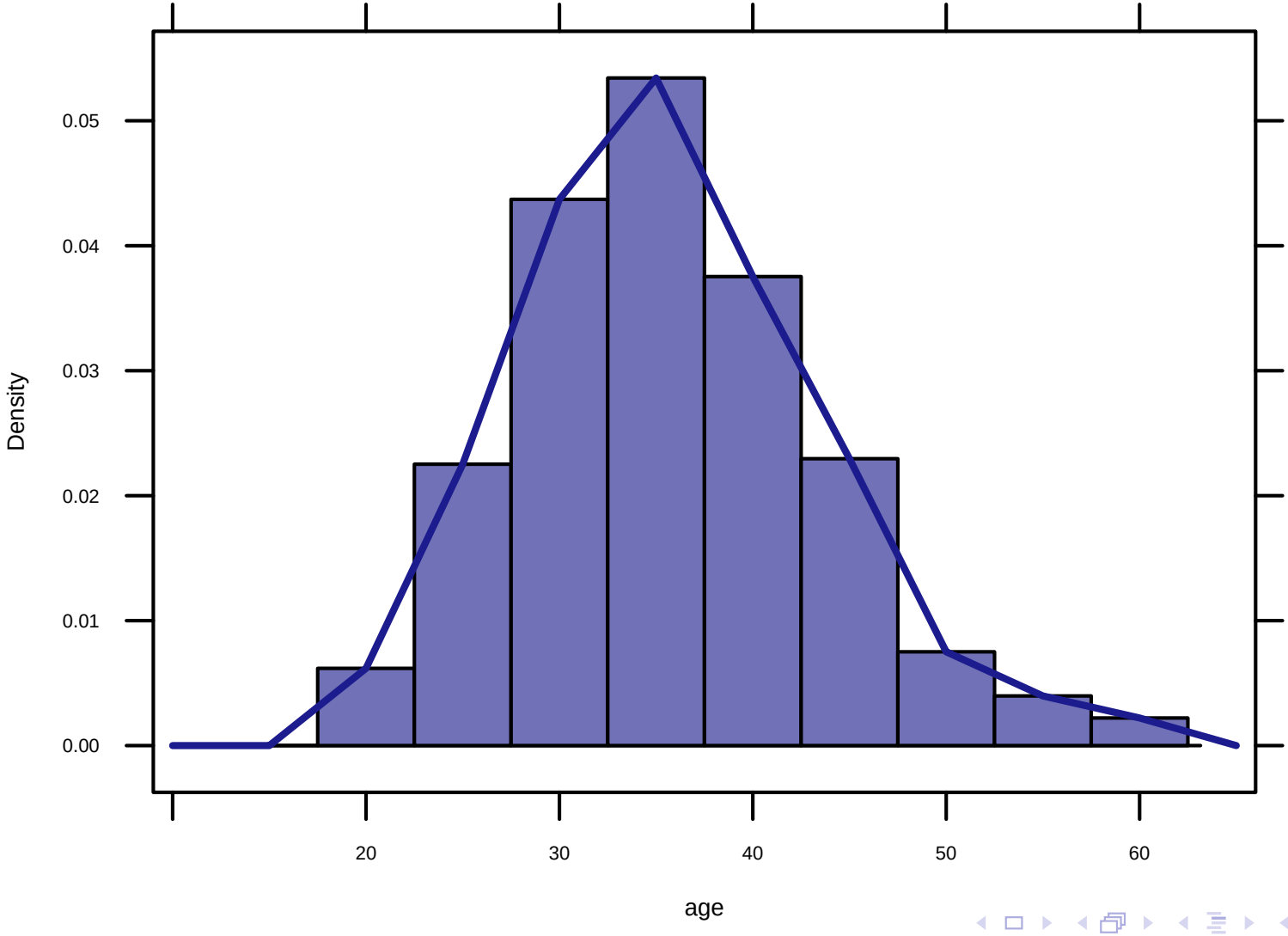
1 Quantitative Variable – Smoother "Histograms"

```
> densityplot(~age, data = HELPrct)  # lattice
> freqpolygon(~age, data = HELPrct)  # mosaic
```



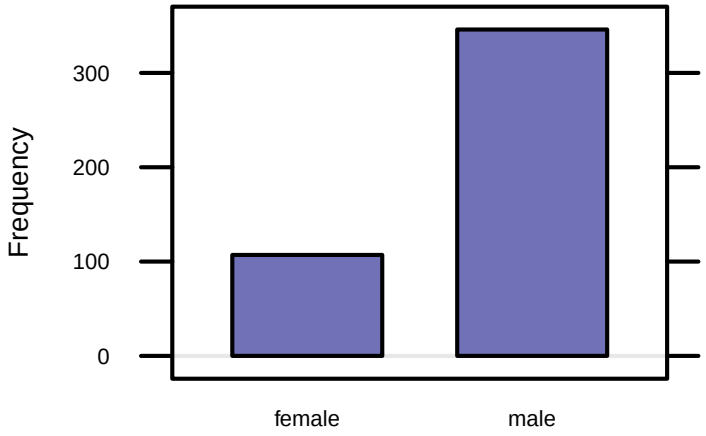


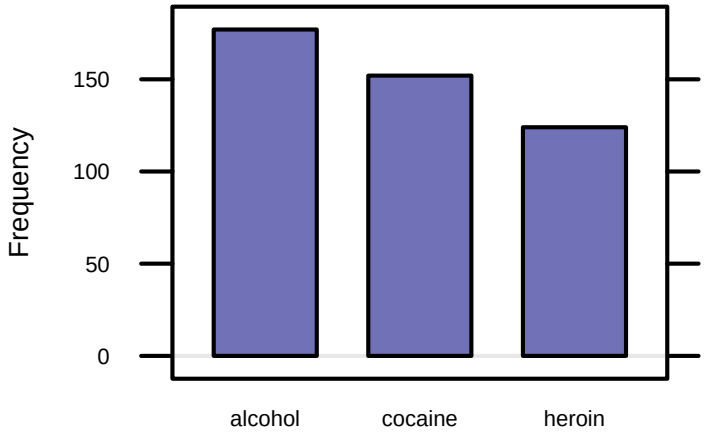
1 Quantitative Variable – Frequency Polygons



1 Categorical Variable

```
> bargraph(~sex, data = HELPrct)  # mosaic
> bargraph(~substance, data = HELPrct)  # mosaic
```





The `barchart()` function in the `lattice` package requires that you first summarize the data. `bargraph()` does the summarizing and then calls `barchart()` to make the plot.

Numerical Summaries

```
> tally(~substance, data = HELPrct)  # mosaic


alcohol cocaine  heroin   Total
    177     152     124     453

> prop(~substance, data = HELPrct)  # mosaic

alcohol
 0.3907
```

Inference

```
> binom.test(~homeless, data = HELPrct)  # mosaic augmented



Exact binomial test

data:  HELPrct$homeless
number of successes = 209, number of trials = 453,
p-value = 0.1101
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.4147 0.5085
sample estimates:
probability of success
                0.4614
```

Summary

Notice the similarities in all these functions:

```
> histogram(~age, data = HELPrct)
> densityplot(~age, data = HELPrct)
> freqpolygon(~age, data = HELPrct)
> bargraph(~substance, data = HELPrct)
> tally(~substance, data = HELPrct)
> prop(~substance, data = HELPrct)
> binom.test(~homeless, data = HELPrct)
```

Here's the pattern to learn:

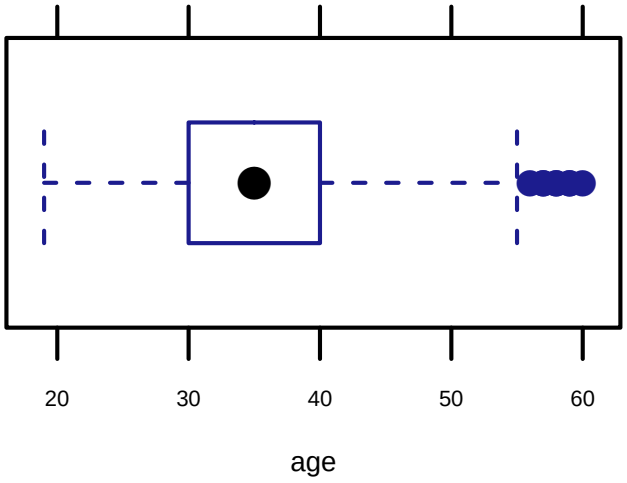
```
> functionname(~variablename, data = ...)
```

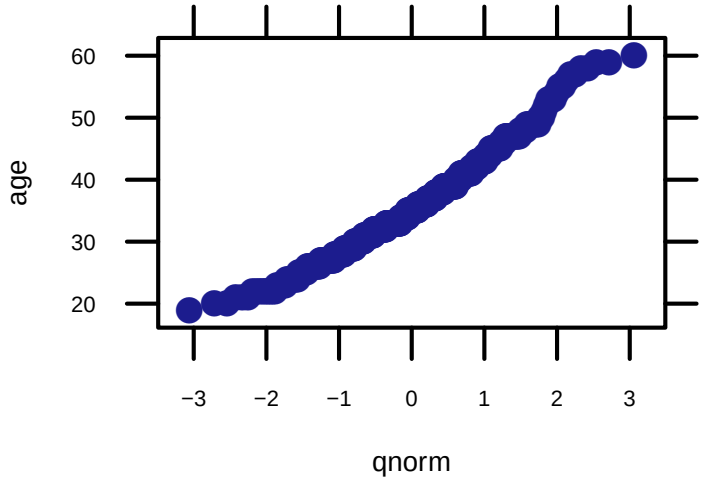
Your turn:

- Create a few plots for some other variables.
- Guess how you would make boxplots and normal-quantile plots.

1 Quantitative Variable – Boxplots and QQ Plots

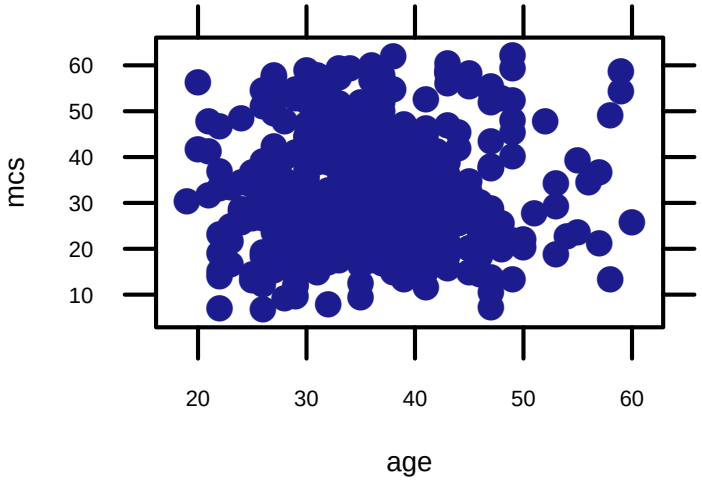
```
> bwplot(~age, data = HELPrct)  # lattice
> qqmath(~age, data = HELPrct)  # lattice
```

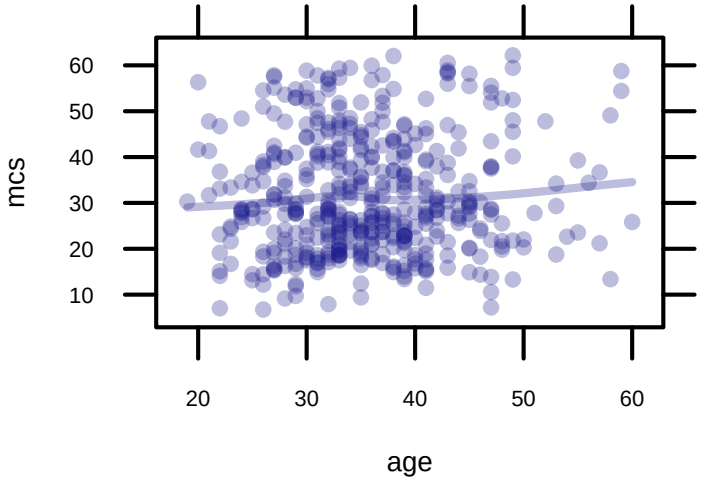




2 Quantitative Variables – Scatter plots

```
> xyplot(mcs ~ age, data = HELPrct)
> xyplot(mcs ~ age, data = HELPrct, alpha = 0.3, cex = 0.5,
+     type = c("p", "smooth"))
```



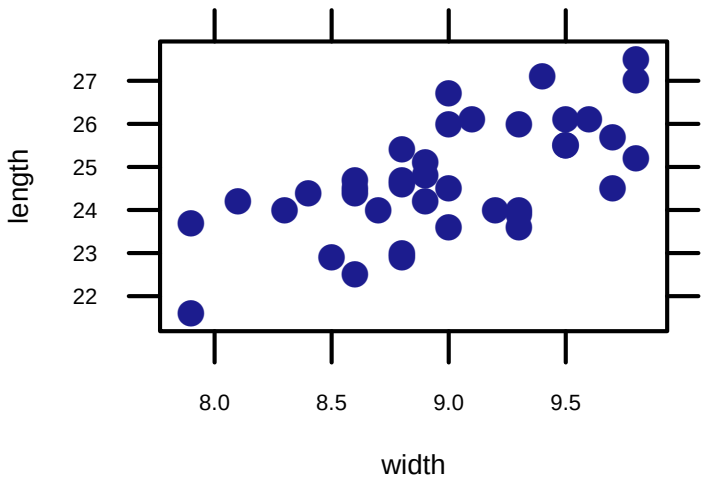


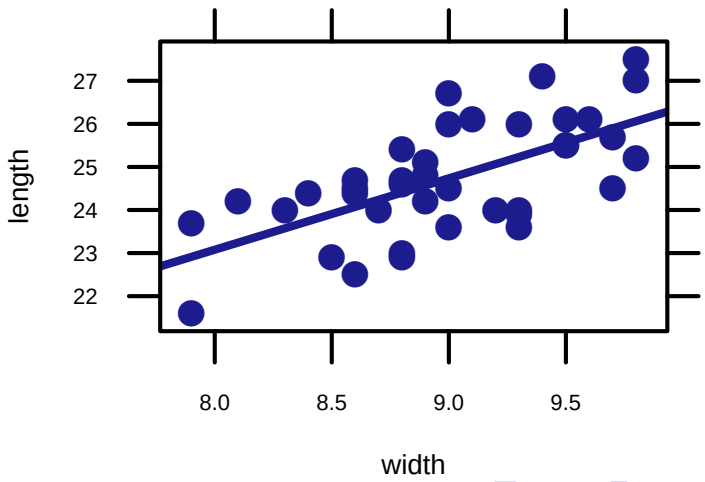
2 Quantitative Variables – Scatter Plots

```
> head(KidsFeet, 2)


   name birthmonth birthyear length width sex biggerfoot domhand
1 David          5        88   24.4   8.4   B          L       R
2  Lars         10        87   25.4   8.8   B          L       L
```

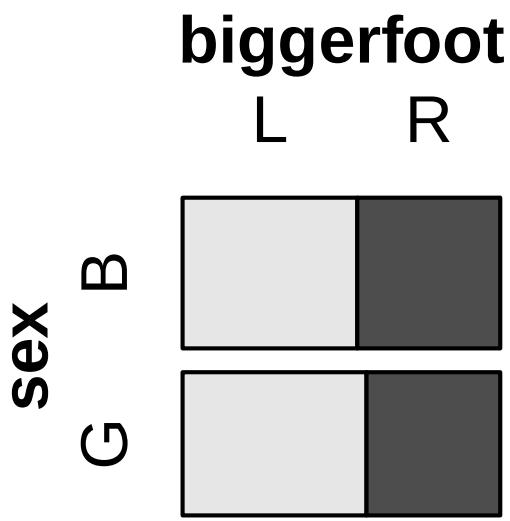
```
> xyplot(length ~ width, data = KidsFeet)
> xyplot(length ~ width, data = KidsFeet, type = c("p", "r"))
```

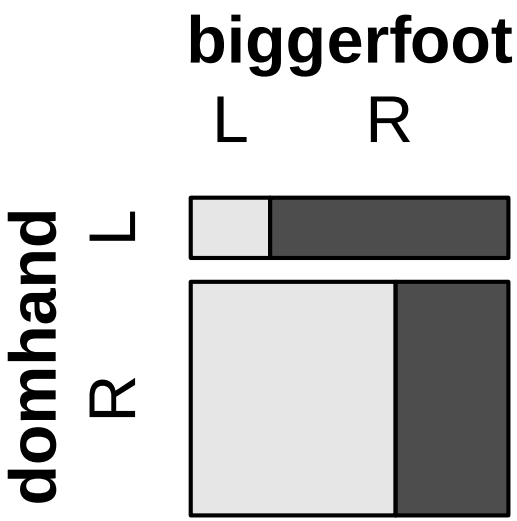




2 Categorical Variables – Mosaic Plots

```
> require(vcd)
> mosaic(biggerfoot ~ sex, KidsFeet)
> mosaic(biggerfoot ~ domhand, KidsFeet)
```

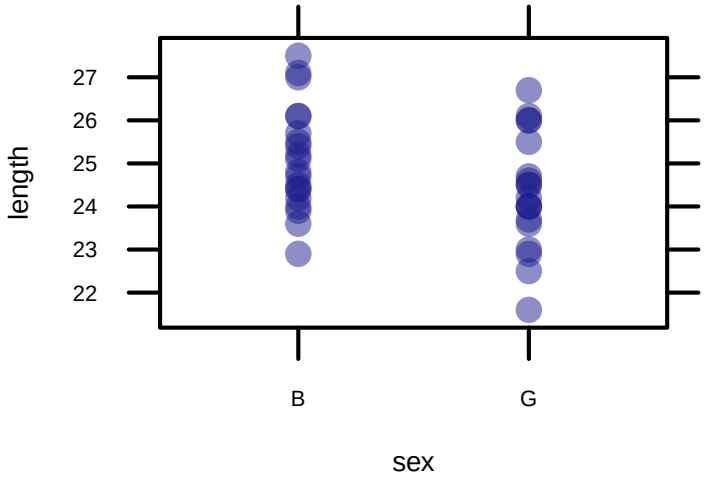


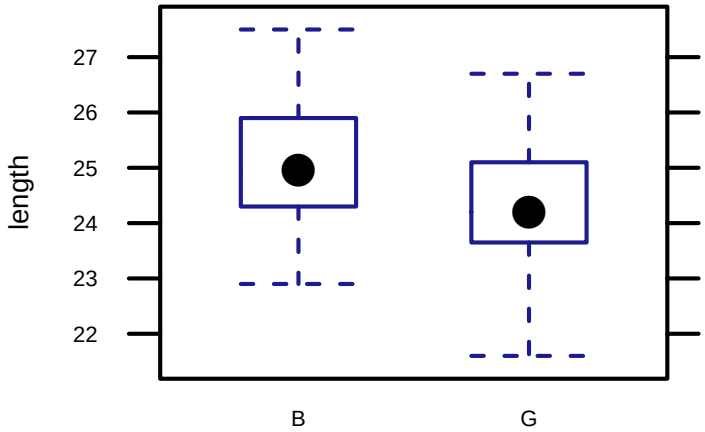


Note: This is not a `lattice` plot, but it does use the formula syntax.

1 Quantitative, 1 Categorical

```
> xyplot(length ~ sex, KidsFeet, alpha = 0.5)
> bwplot(length ~ sex, KidsFeet)
```

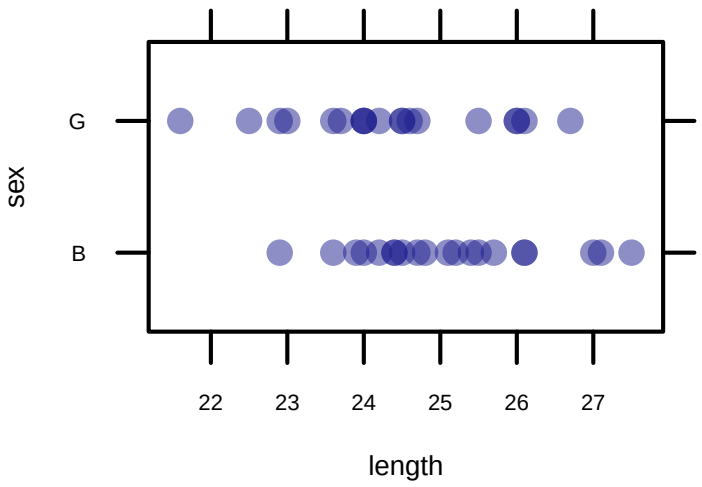


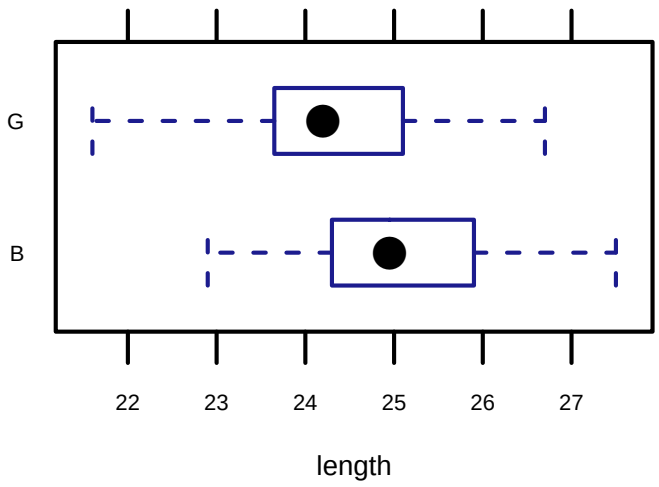


Q: What happens if we reverse the positions of `length` and `sex`?

1 Categorical, 1 Quantitative

```
> xyplot(sex ~ length, KidsFeet, alpha = 0.5)
> bwplot(sex ~ length, KidsFeet)
```





Alternative: Treat categorical variable like a covariate (stay tuned).

Your Turn

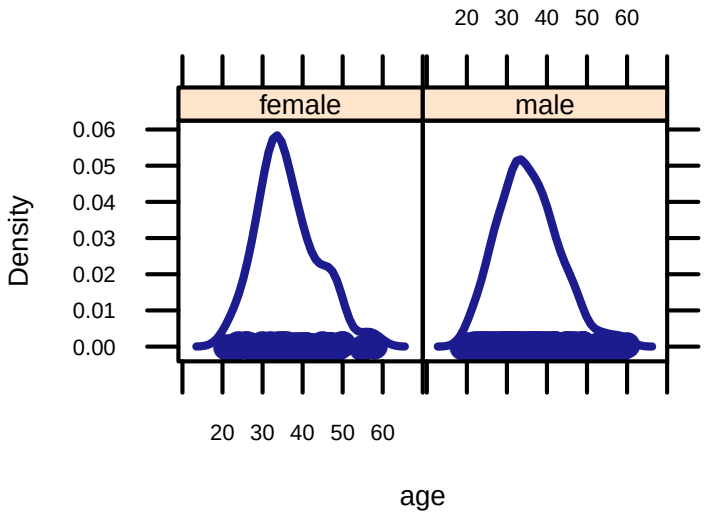
Q: How would you do each of the following?

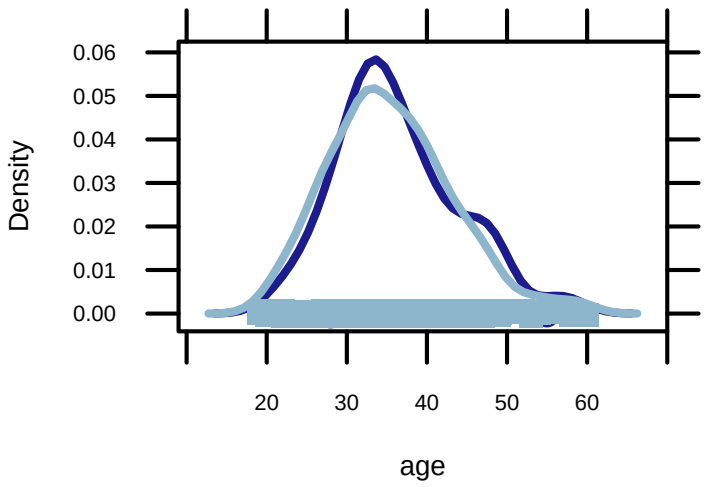
1. Create a plot to compare the ages of the three substance groups in the HELP study.
2. Compute the mean and median age for each of the three substance groups.
3. Tally the number of kids in each sex-handedness combination.
4. Create a plot to see how the length of a kid's foot depends on the kid's age.

Adding Covariates

Lattice provides two additional ways of adding in additional variables: conditioning (multi-panel plots) and grouping (multiple overlaid plots).

```
> densityplot(~age | sex, data = HELPrct)
> densityplot(~age, groups = sex, data = HELPrct, lty = 1)
```

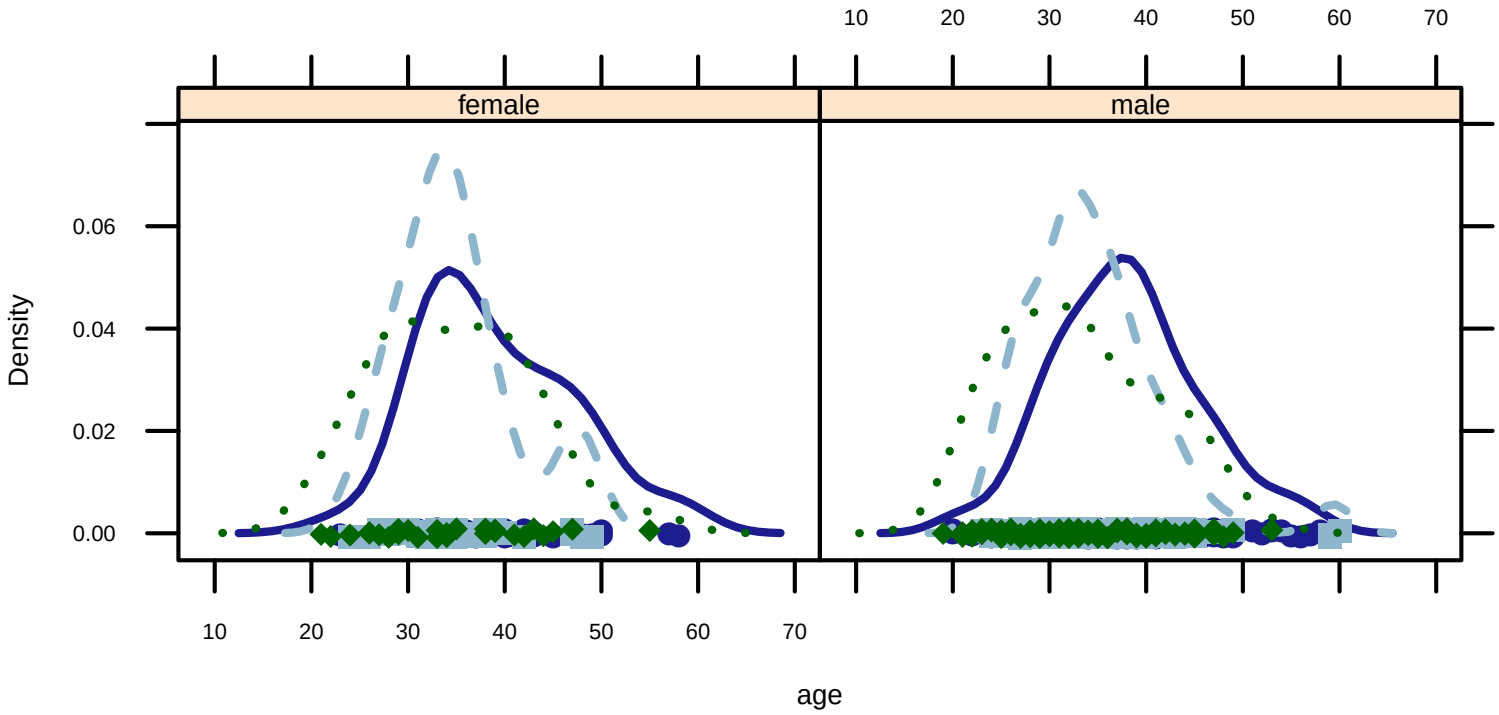




Adding Covariates

We can do both conditioning (multi-panel plots) and grouping (multiple overlaid plots) in the same plot.

```
> densityplot(~age | sex, groups = substance, data = HELPrct)
```



Numerical Summaries

This syntax is useful for numerical summaries as well.

```
> tally(biggerfoot ~ sex, KidsFeet)


          sex
biggerfoot      B      G
     L     0.5500 0.5789
     R     0.4500 0.4211
     Total 1.0000 1.0000


> tally(~biggerfoot | sex, KidsFeet)


          sex
biggerfoot      B      G
     L     0.5500 0.5789
     R     0.4500 0.4211
     Total 1.0000 1.0000
```

Numerical Summaries

```
> tally(~biggerfoot | sex, KidsFeet, format = "count")

          sex
biggerfoot  B  G
     L     11 11
     R      9  8
     Total 20 19

> tally(biggerfoot ~ domhand | sex, KidsFeet)

       sex
domhand      B      G
  L     0.2500 0.1579
  R     0.7500 0.8421
  Total 1.0000 1.0000
```

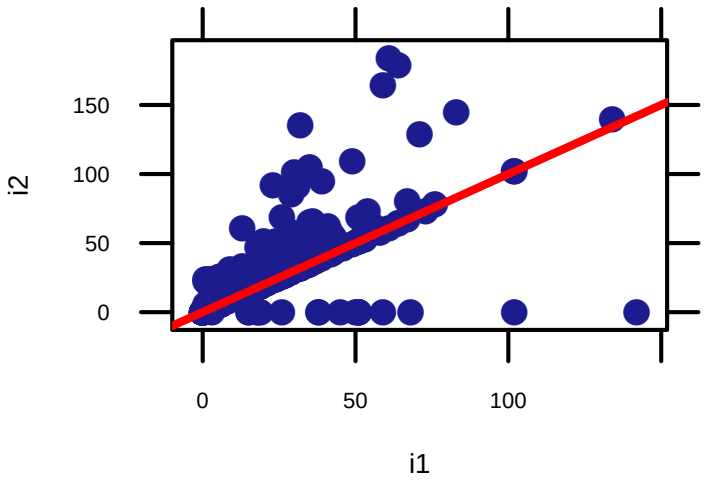
Your Turn

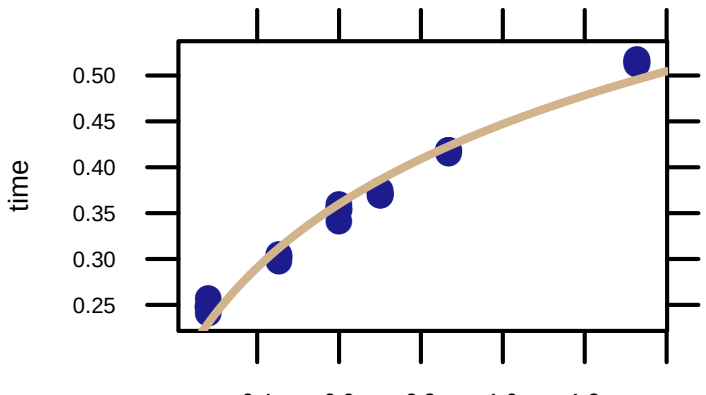
1. The `i1` and `i2` variables in `HELPrct` give the average and maximum number of drinks per day (standard units) consumed in the 30 days prior to admission.
   1.1 How do the distributions compare among the three substance groups?
   1.2 Use a scatter plot to uncover a problem with this data.
2. Create an interesting plot using the `iris` data set.

Stacking Things Up

The `mosaic` package contains some functions that make it easier to stack things up on a plot.

```
> xyplot(i2 ~ i1, data = HELPrct)
> plotFun(x ~ x, add = TRUE, col = "red")
```



Stacking Things Up

```
> require(fastR)
> model <- lm(time ~ log(height), data = balldrop)
> coef(model)


(Intercept) log(height)
     0.4471      0.1708


> fit <- makeFun(model)
> xyplot(time ~ height, balldrop)
> plotFun(fit(height) ~ height, add = TRUE, lwd = 2,
+     col = "tan")
```