manipulate

Biostatistics 140.776

Interactivity

- RStudio provides two mechanisms for introducing interactivity in statistical models
- **manipulate**: A simple framework for making interactive plots; only works in RStudio
- **shiny**: A much more complex and feature-filled framework for developing and deploying interactive web apps

manipulate

- R package (on CRAN)
- Simple mechanism for adding sliders, checkboxes and buttons to plots
- Only usable with base graphics (i.e. `plot()`)
- Plot calls are wrapped in a call to the `manipulate()` function
- A little “wheel/gear” appears in upper corner of plot that will toggle controls

Manipulate controls

- **Slider**: Choose from a “continuous” range of values
- **Picker**: Pick from a “drop down menu”; only select a single thing
- **Checkbox**: Select from a few different categories of a variable (can select more than one)
- **Button**: Trigger an action (good for simulations)
- These can be used in combination

manipulate: Slider

```
library(manipulate)
library(ggplot2)

data <- read.csv("eno.csv")
manipulate(
        qplot(log(eno), data = data, bins = n.breaks),
        n.breaks = slider(3, 20)
)
```

plotting expression

variable to control

controller

manipulate: Picker

```
## Merge allergic sensitivity data
skin <- read.csv("data/skin.csv")
m <- merge(data, skin, by = "id")

manipulate(
        qplot(log(eno),
              data = filter(m, mopos == allergic),
              bins = n.breaks),
        n.breaks = slider(3, 20, label = "Bins"),
        allergic = picker("Yes" = "yes", "No" = "no",
                          label = "Mouse Allergic?")
)
```

Subset the data based on variable

manipulate: Combining controls

```
## Load/merge the data
eno <- read.csv("eno.csv")
env <- read.csv("environmental.csv")
skin <- read.csv("skin.csv")
m <- merge(eno, env, by = "id")
m <- merge(m, skin, by = "id")

## Precompute ranges
xlim <- range(log(m$pm25), na.rm = TRUE)
ylim <- range(log(m$eno), na.rm = TRUE)
```

manipulate: Combining controls

Subset data

```
manipulate({
        g <- ggplot(data = filter(m, mopos == allergic),
                    aes(log(pm25), log(eno))) +
                xlim(xlim) + ylim(ylim) +
                geom_smooth(method = "loess", span = span.p)
        if(addpoints)
                g <- g + geom_point()
        print(g)
}, allergic = picker("Yes" = "yes", "No" = "no",
                     label = "Mouse Allergic?"),
addpoints = checkbox(FALSE, "Add Points?"),
span.p = slider(0.2, 1, initial = 2/3, label = "Span"))
```

Show points

Setup smoother

manipulate: Buttons

```
manipulate({
        if(reset) {
                seed <- as.integer(Sys.time())
                set.seed(seed)
        }
        x <- rnorm(100)
        y <- 4 + 1.5 * x + rnorm(100)
        qplot(x, y, xlim = c(-4, 4), ylim = c(-1, 10))
}, reset = button("Reset seed?"))
```

Summary

- manipulate requires the use of RStudio
- Doesn’t allow you to “deploy” your visualization in any useful way
- Interactivity is limited (sliders, checkboxs, menus, and buttons)
- But, a good quick and dirty solution!