Advanced Graphics in R

Laurel Stell

February 7, 2018

Stanford MEDICINE | Data Studio

Introduction

R Markdown file and slides

Download in 3 easy steps:

1. http://web.stanford.edu/~lstell/
2. Click on "Data Studio presentation: Advanced graphics in R" at bottom of page
3. Click on links under "Background readings"

Objectives

- Examples demonstrating the benefits of **lattice** or **ggplot2** package
- Help you decide which to try
- Introduce keywords so that you will be able to find additional help in R documentation and by Googling
- If you already use one of these packages, you might still learn something new—or decide to use the other one as well.
- Also **ComplexHeatmap** package
- Not a tutorial

Prerequisites

Basic knowledge of R:

- Factors, data frames, etc
- Defining functions; named arguments, optional arguments
- Installing and loading packages
- Base graphics functions such as **plot**

Caveats

- These packages are based on **grid** package. Do *not* mix with base graphics such as par(), split.screen(), axis(), legend().
- Simultaneously loading both **lattice** and **ggplot2** into R (or your brain) might lead to errors.
  - Probably best to choose one or the other initially
  - This presentation probably biased towards **lattice** because that is what I primarily use—only because I was introduced to it first.

Packages used

```
library(plyr)
library(reshape2)
library(latticeExtra)
library(ggplot2)
library(ComplexHeatmap)
```

Pedagogical approach

- Four examples
  - **lattice** solution
  - **ggplot2** solution
- **ComplexHeatmap** example
- Ask questions any time

Multiple box plots for multiple y variables

iris data set

The **iris** data frame gives the measurements in centimeters for 50 flowers from each of 3 species for the following:

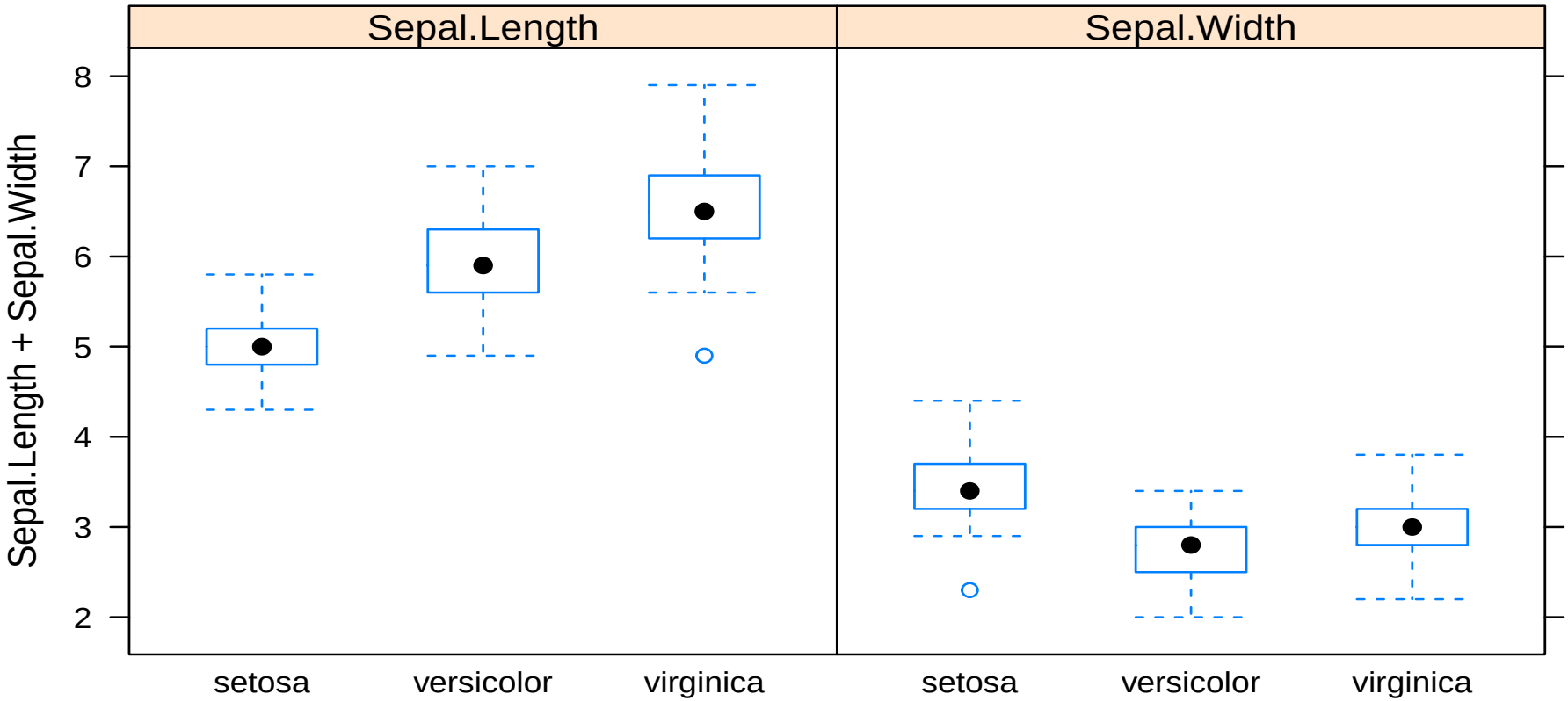
```
data(iris)
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length"
## [4] "Petal.Width"  "Species"
```

```
# Will need ID for each flower later
iris <- transform(iris, ID=seq_len(nrow(iris)))
```

Simple box plots with **lattice**

```
bwplot(Sepal.Length + Sepal.Width ~ Species, iris, outer=T)
```



Desired improvements

- Use different y-axis scales in two panels
- y-axis label is redundant
- Clean up other labels
- What if you wanted to plot 10 variables this way?

String and factor manipulation

See Rmd file for how to

- Capitalize species names
- Apply that change to factor levels
- Replace period with space in strings

It's complicated ... and really doesn't have to do with graphics.

Digression on **reshape2**

Use for either **lattice** or **ggplot2**

```
df.long <- melt(df.short, id.vars="ID")
```

df.short

ID	X	Y	Z
A1	3	1	5
B3	2	0	4

→

df.long

ID	variable	value
A1	X	3
A1	Y	1
A1	Z	5
B3	X	2
B3	Y	0
B3	Z	4

(**tidyr** is an "evolution" of **reshape2**)

Apply **melt** to **iris**

```
names(iris)
```

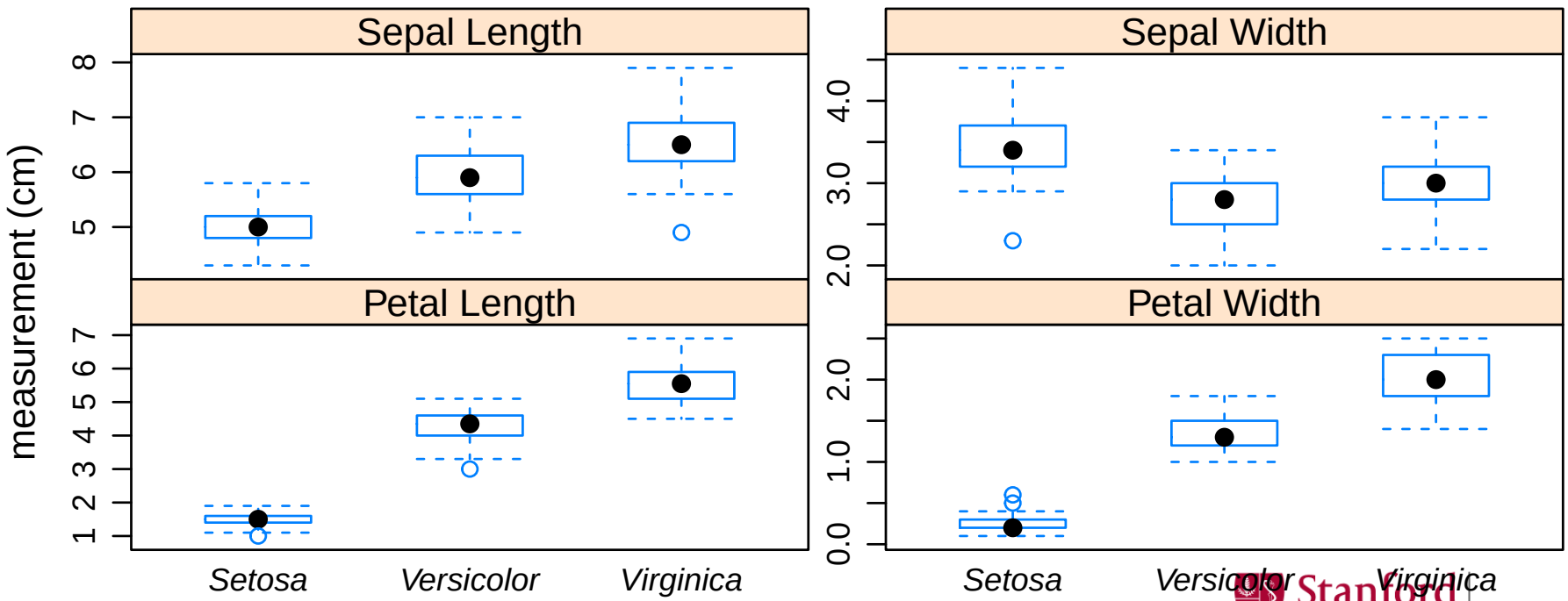
```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length"
## [4] "Petal.Width"  "Species"      "ID"
```

```
iris.long <- melt(iris, id.vars=c("ID","Species"),
                  variable.name="measure")
names(iris.long)
```

```
## [1] "ID"      "Species" "measure" "value"
```

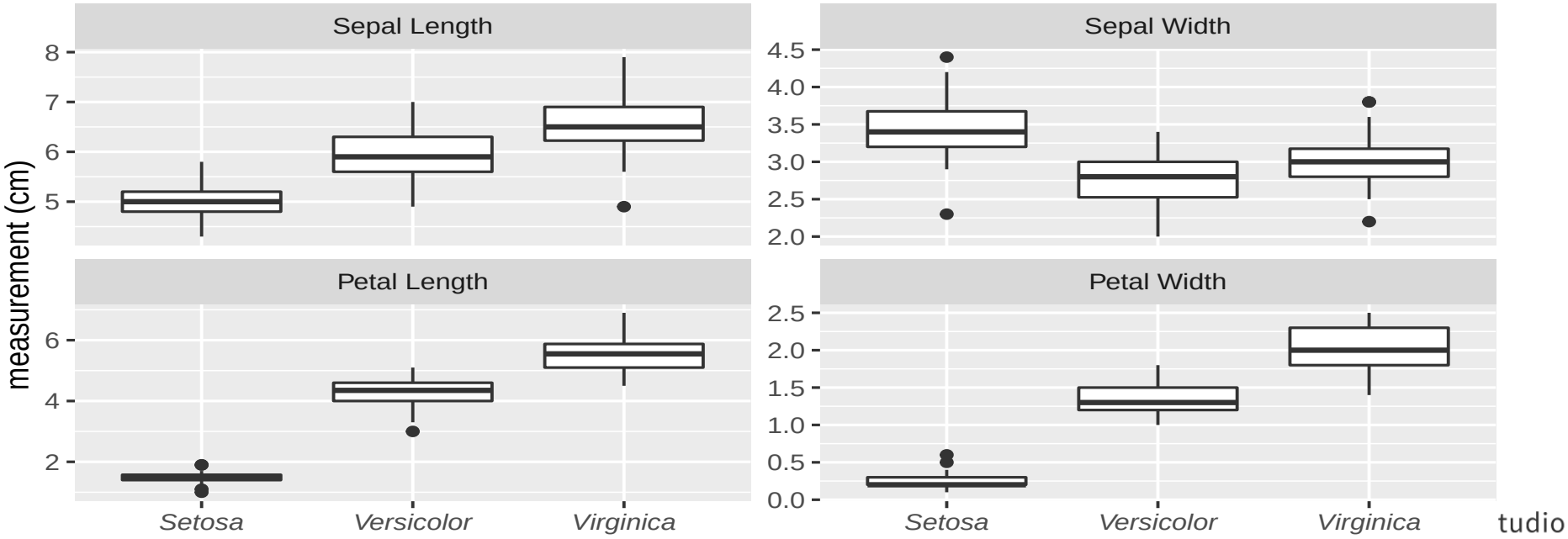
Improved **lattice** box plots

```
bwplot(value ~ Species | measure, iris.long,
       as.table=T, layout=c(2,2),
       scales=list(x=list(font=3), y=list(relation="free")),
       ylab="measurement (cm)",
       strip=strip.custom(factor.levels=lab.panel))
```



ggplot2 box plots

```
names(lab.panel) <- levels(iris.long$measure)
ggplot(iris.long, aes(x=Species, y=value)) +
  geom_boxplot() +
  facet_wrap(~ measure, ncol=2, scales="free_y",
             labeller=labeller(measure=lab.panel)) +
  xlab("") + ylab("measurement (cm)") +
  theme(axis.text.x=element_text(face="italic"))
```



tudio

Scatter plots with regression lines for groups of samples

Relationship between length and width of flower parts

Need separate plots for sepal and petal.

- Split measurement name:
  - **part**: Sepal or Petal
  - **dim**: Length or Width
- Drop **measure**

See Rmd file.

Another digression on **reshape2**

- **dcast** undoes **melt** . . . if it can determine which rows belong together
- This was reason for adding **ID**.

```
print(subset(iris.med, ID %in% 1:2), row.names=F)
```

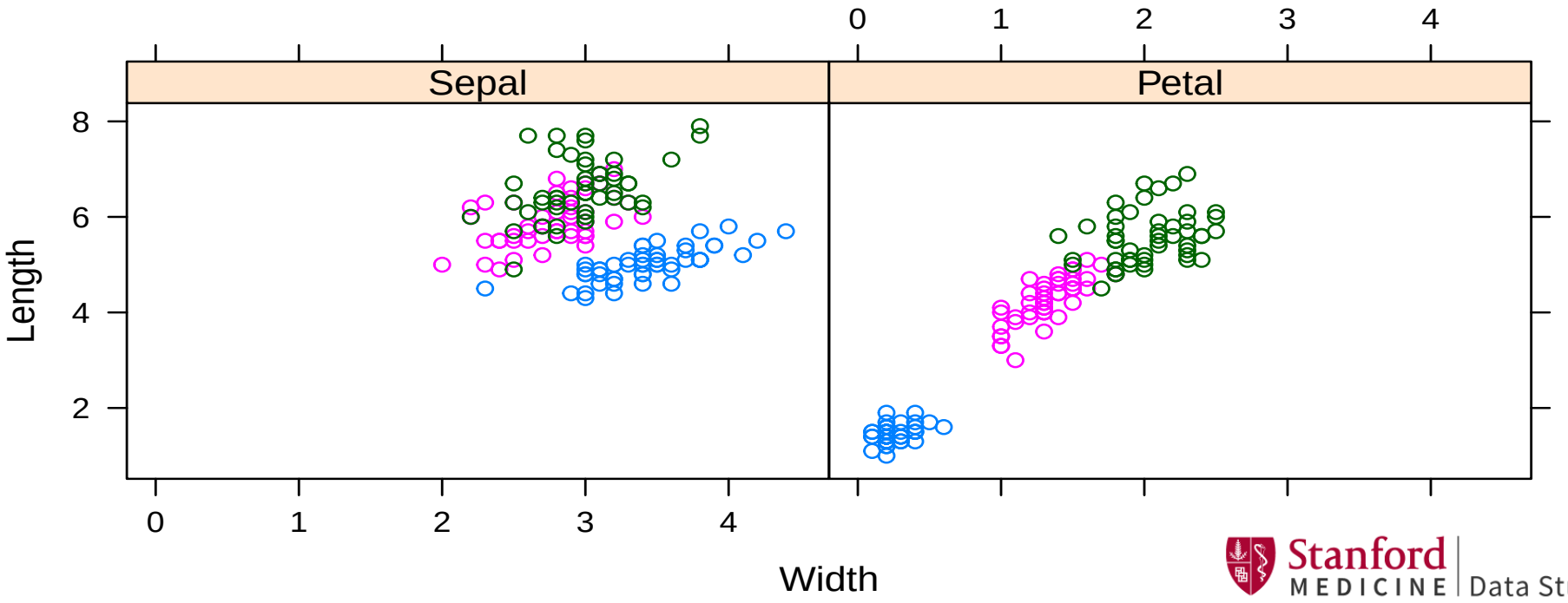
```
##  ID Species value  part    dim
##   1  Setosa   5.1 Sepal Length
##   2  Setosa   4.9 Sepal Length
##   1  Setosa   3.5 Sepal  Width
##   2  Setosa   3.0 Sepal  Width
##   1  Setosa   1.4 Petal Length
##   2  Setosa   1.4 Petal Length
##   1  Setosa   0.2 Petal  Width
##   2  Setosa   0.2 Petal  Width
```

Relationship betwen length and width (basic version)

```
iris.med <- dcast(iris.med, ... ~ dim, value.var="value")
names(iris.med)
```

```
## [1] "ID"      "Species" "part"    "Length"  "Width"
```

```
xyplot(Length ~ Width | part, iris.med, groups=Species)
```



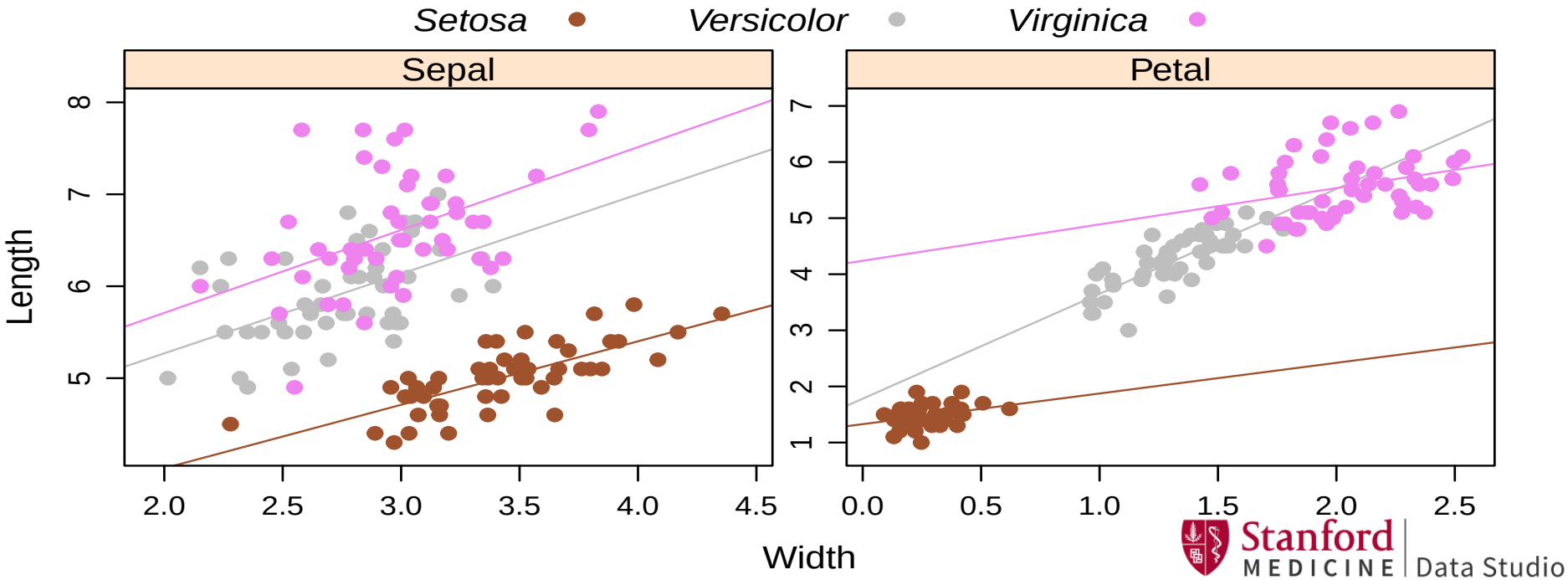
Desired improvements

- Jitter (300 rows in iris.med, but only 219 unique marker locations)
- Regression lines
- Filled markers
- Legend
- Different scales on both axes
- Change colors

```
col.iris <- c("sienna", "gray", "violet")
```

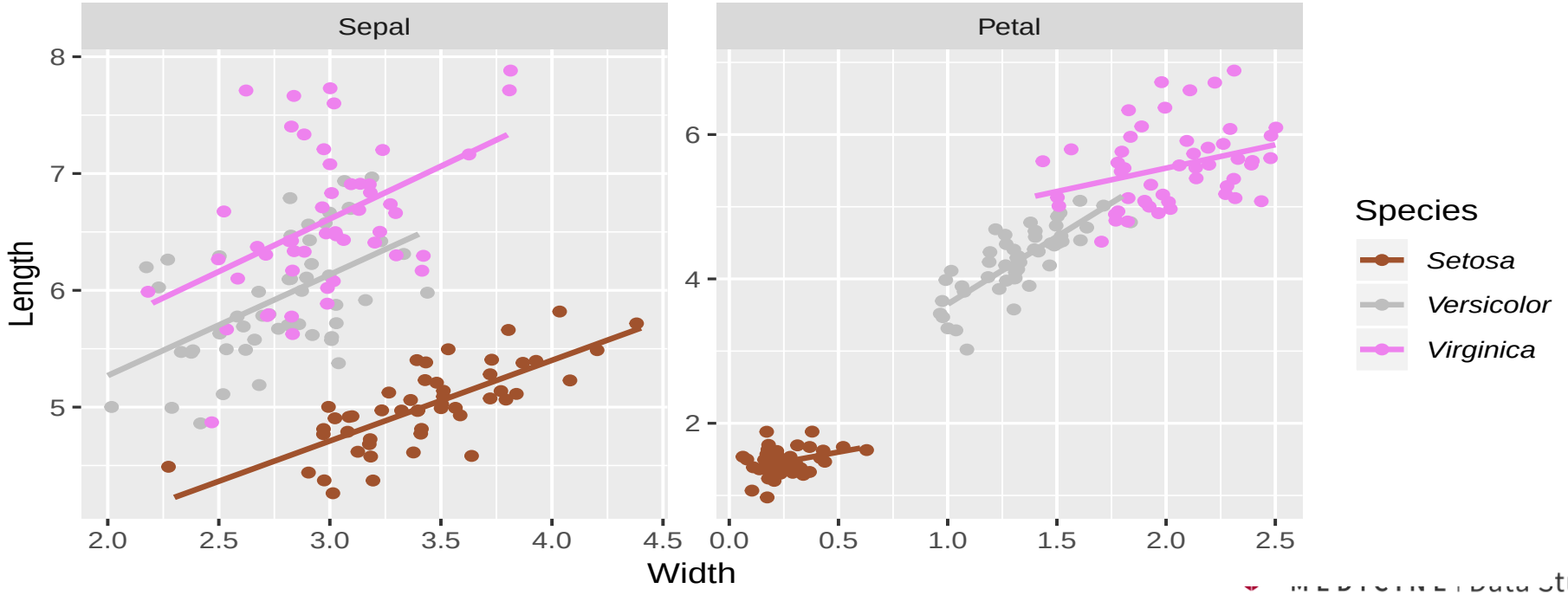
lattice version

```
xyplot(Length ~ Width | part, iris.med, type=c("p","r"),
       groups=Species, auto.key=list(columns=3, font=3),
       scales=list(relation="free"), jitter.x=T, amount=0.05,
       par.settings=list(superpose.symbol=list(pch=19,
                                               col=col.iris),
                         superpose.line=list(col=col.iris)))
```



ggplot2 version

```
ggplot(iris.med, aes(x=Width, y=Length, color=Species)) +
  geom_jitter() +
  scale_color_manual(values=col.iris) +
  geom_smooth(method="lm", se=F) +
  facet_wrap(~ part, scales="free") +
  theme(legend.text=element_text(face="italic"))
```

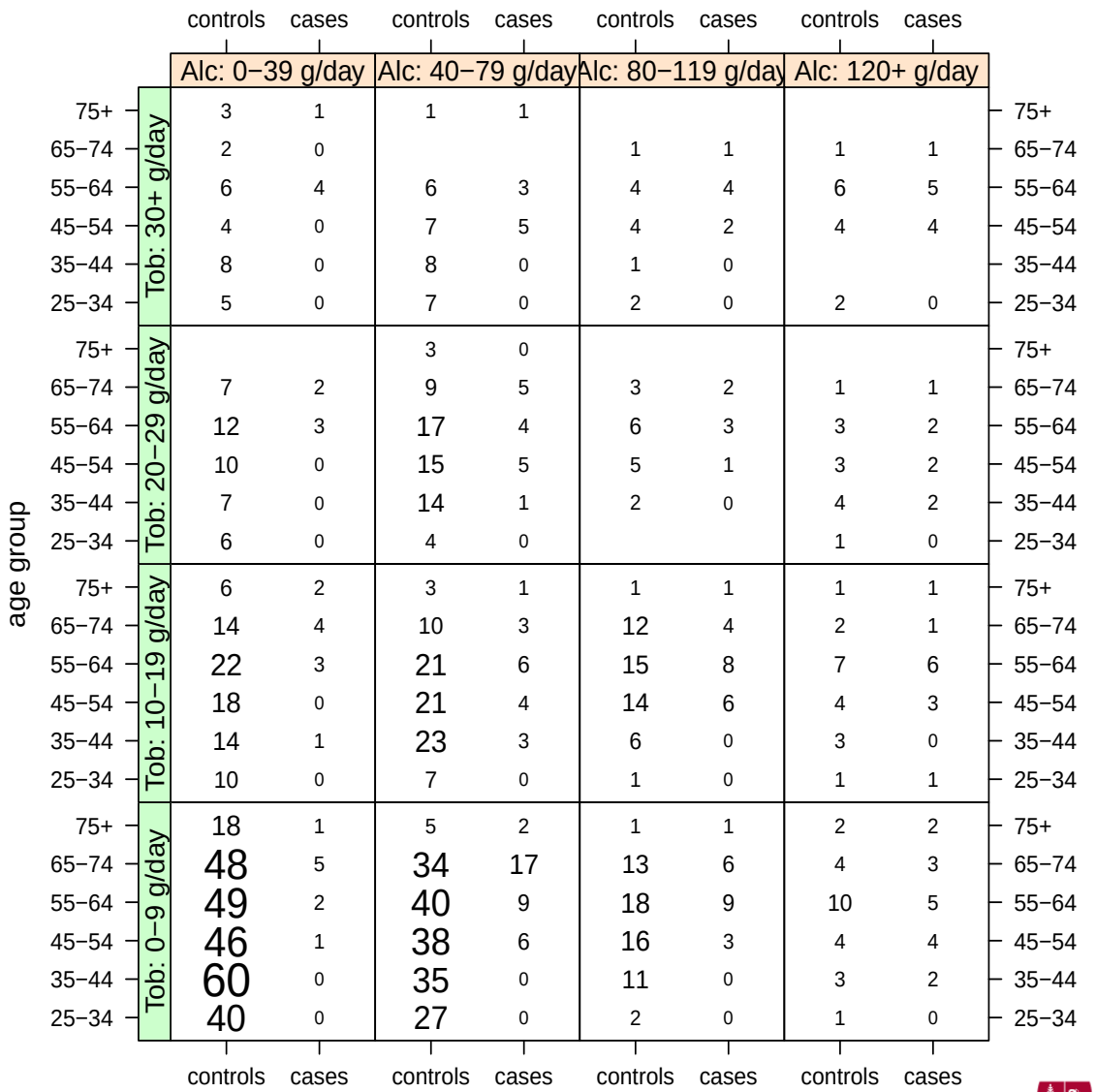


Plotting text; panels based on two variables

esoph data set

- Case-control study of esophageal cancer
- Number of cancer cases and controls for:
  - 6 age groups
  - 4 levels of alcohol consumption
  - 4 levels of tobacco consumption
- 3 independent variables, 2 dependent variables
- Commonly visualized with mosaic plot, but ...

The plot (**lattice** version)



Closer look at original data frame

```
names(esoph)
```

```
## [1] "agegp"     "alcgp"     "tobgp"     "ncases"
## [5] "ncontrols"
```

```
levels(esoph$alcgp)
```

```
## [1] "0-39g/day" "40-79"     "80-119"    "120+"
```

```
levels(esoph$tobgp)
```

```
## [1] "0-9g/day" "10-19"    "20-29"    "30+"
```

Approach: data manipulation

- Put numbers of cases and controls in single column in data frame

```
esoph.long <- melt(esoph,
                   measure.vars=c("ncases","ncontrols"),
                   variable.name="status", value.name="n")
```

```
names(esoph.long)
```

```
## [1] "agegp"  "alcgp"  "tobgp"  "status" "n"
```

- Rmd file has code for getting better group names; put in **grps.alc**, **grps.tob**, **grps.status**

Approach: **lattice** graphics

- Each high-level **lattice** function calls another function to actually create plot for each panel
  - By default, **xyplot** calls **panel.xyplot**, for example
  - Different function can be specified in high-level call
  - For example, `panel=MyPanelFn`
- What objects can **MyPanelFn** use?
  - **args()** or default function's help page show formal parameters that will be passed to it
  - Can also use objects in environment that called high-level function
    -  and  but also  and 
    - See explanation of R's lexical scope for details, eg cran.r-project.org/doc/manuals/r-release/R-intro.html#Scope

My panel function

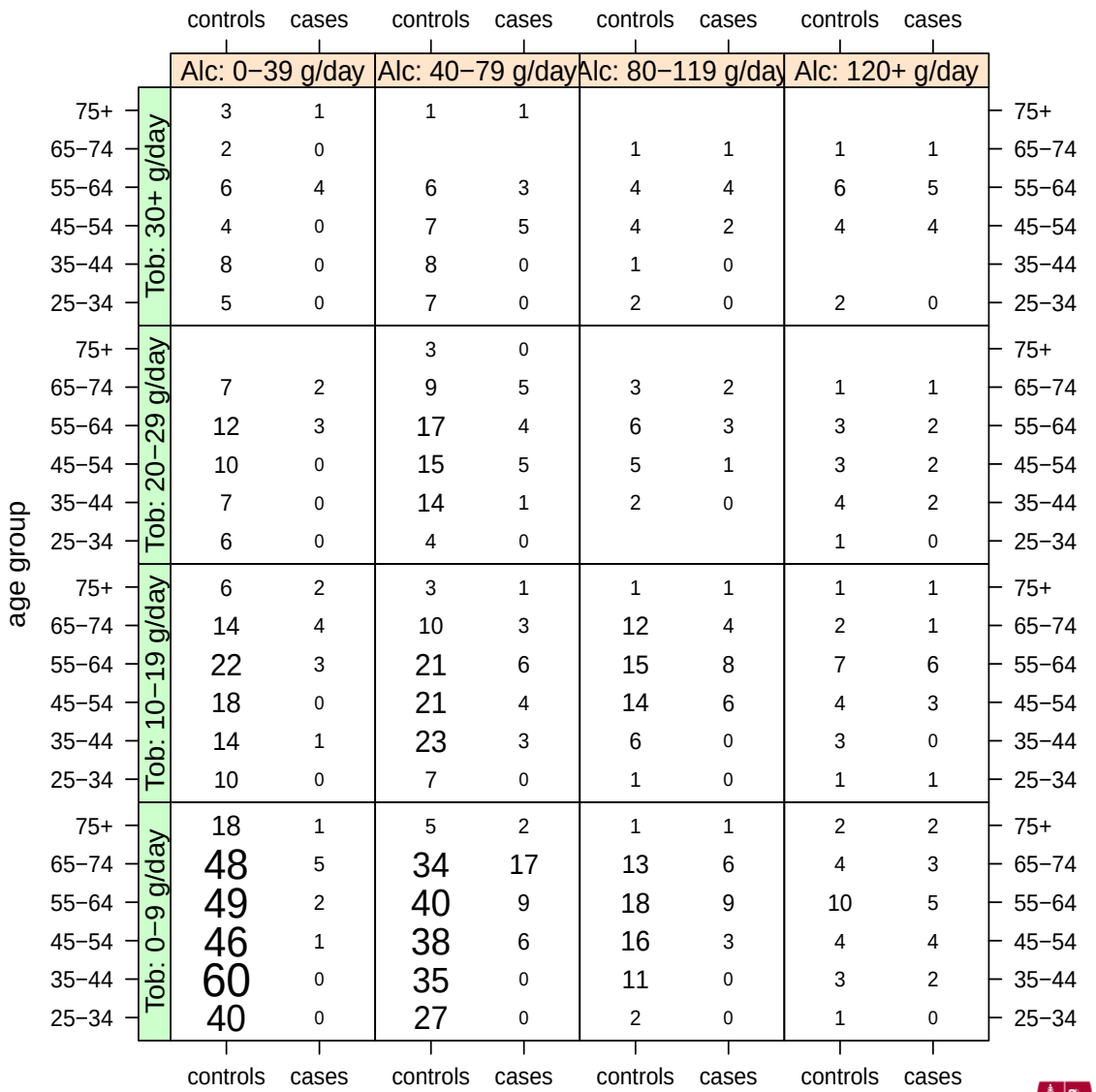
```
fac <- 1/max(esoph.long$n)
MyPanelFn <- function(x, y, subscripts, ...) {
  n <- esoph.long[subscripts, "n"]
  panel.text(x=x, y=y, lab=n, cex=0.7 + fac*n)
}
```

- `subscripts` contains row indices of `x` and `y` data for that panel in original data frame
- Use it to get the numbers to "plot"

lattice code

```
h <- xyplot(agegp ~ status | alcgp * tobgp, esoph.long,
            scales=list(alternating=3,
                        x=list(labels=grps.status)),
            xlab=NULL, ylab="age group",
            panel=MyPanelFn)
h <- useOuterStrips(h,
                    strip=strip.custom(
                              factor.levels=grps.alc),
                    strip.left=strip.custom(
                              factor.levels=grps.tob))
```

lattice version's encore



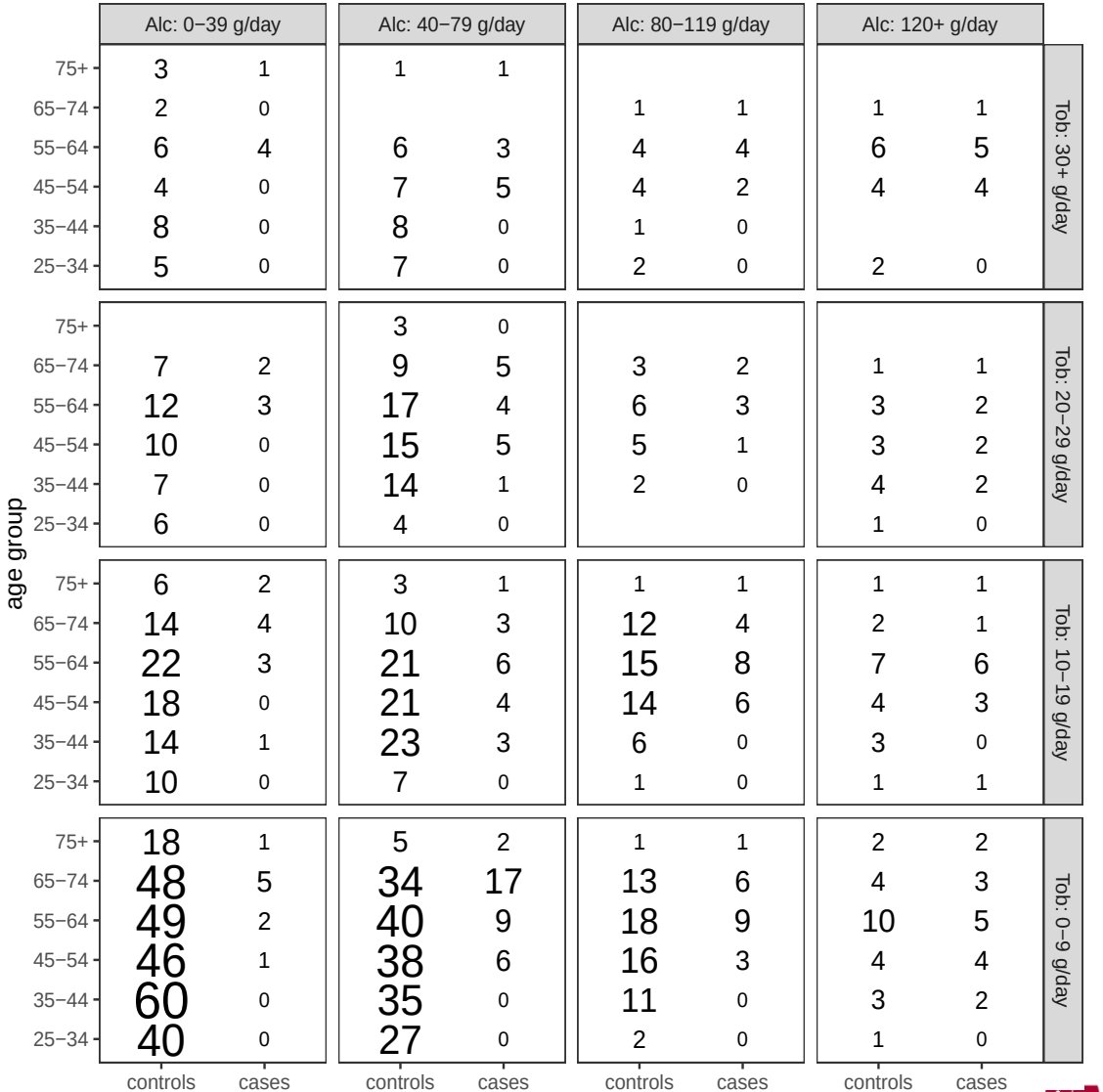
Stanford MEDICINE | Data Studio

ggplot2 code

```
h <- ggplot(esoph.long, aes(status, agegp, label=n)) +
  geom_text(aes(size=n)) +
  scale_size(range=c(3,8), guide=F) +
  facet_grid(tobgp ~ alcgp, as.table=F,
             labeller=labeller(tobgp=grps.tob,
                               alcgp=grps.alc)) +
  xlab("") + ylab("age group") +
  scale_x_discrete(labels=grps.status) +
  theme_bw() + theme(panel.grid.major=element_blank())
```

For discrete values, cannot put tick labels on both sides.

ggplot2 version



Scatter plots with color and shape determined by different variables

Parameters for both **lattice** and **ggplot2**

```
col.tob <- c("pink1", "indianred", "gray50", "black")
lab.tob <- sub("g/day", "", levels(esoph$tobgp))
lab.alc <- sub("g/day", "", levels(esoph$alcgp))
lab.age <- function(string) paste("age:", string)
```

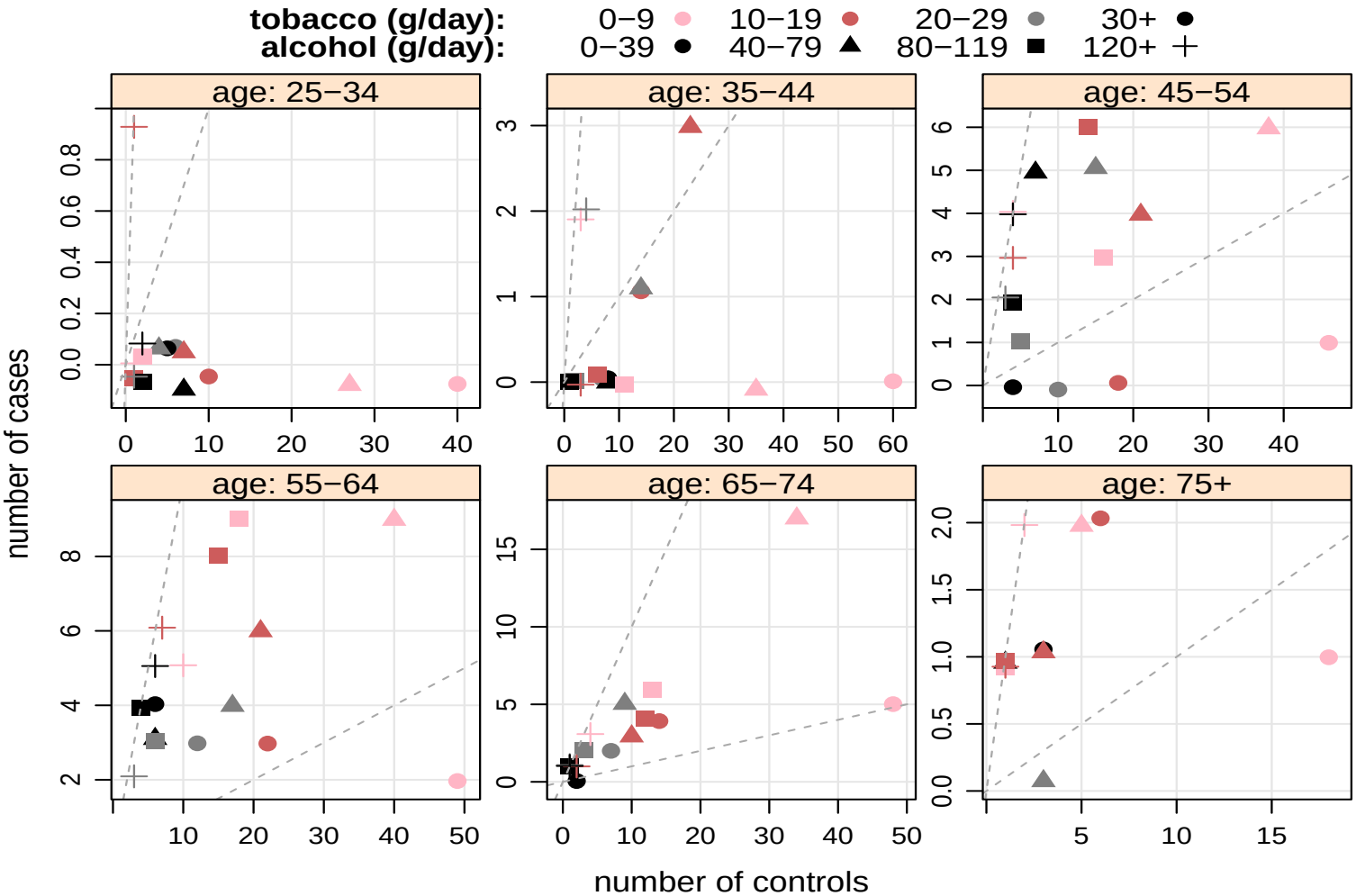
lattice code: definition of panel function and legend

```
pch.alc <- c(16,17,15,3)
MyPanelFn <- function(x, y, subscripts, ...) {
  pch <- pch.alc[as.numeric(esoph$alcgp[subscripts])]
  col <- col.tob[esoph$tobgp[subscripts]]
  panel.xyplot(x, y, pch=pch, col=col, ...)
}
l.key=list(space="top", adj=1, columns=5,
           text=list(c("tobacco (g/day):",
                       "alcohol (g/day):",
                       rbind(lab.tob, lab.alc)),
                     font=c(rep(2,2), rep(1,8))),
           points=list(pch=c(rep(NA, 2),
                             rbind(pch.alc[1], pch.alc)),
                       col=c("white", "white",
                             rbind(col.tob, "black"))),
           between=1,
           between.columns=1)
```

lattice code: **xyplot** and layers

```
h <- xyplot(njtr ~ ncontrols | agegp,
            data=transform(esoph,
                           njtr=jitter(ncases, amount=0.1)),
            as.table=T, grid=T, cex=1.1,
            scales=list(relation="free"),
            xlab="number of controls",
            ylab="number of cases",
            panel=MyPanelFn,
            strip=strip.custom(factor.levels=
                               lab.age(levels(esoph$agegp))),
            key=l.key)
h <- h + latticeExtra::layer(panel.abline(a=0, b=1, lty=2,
                                          col="darkgray"))
h <- h + latticeExtra::layer(panel.abline(a=0, b=0.1, lty=2,
                                          col="darkgray"))
```

lattice version

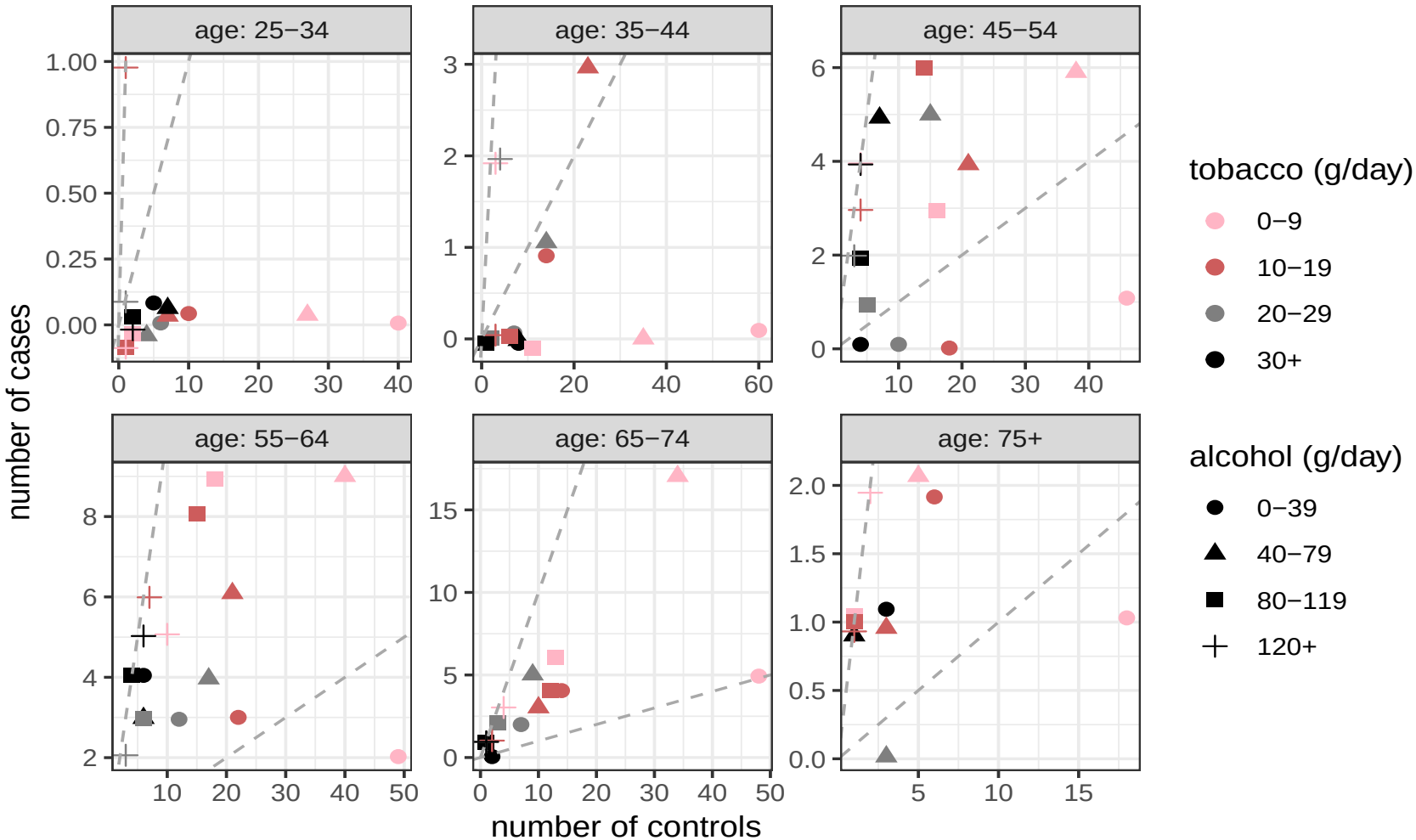


Stanford MEDICINE | Data Studio

ggplot2 code

```
h <- ggplot(esoph,
            aes(x=ncontrols, y=ncases,
                color=tobgp, shape=alcgp)) +
  geom_jitter(width=0, height=0.1, size=2.3) +
  geom_abline(intercept=0, slope=c(1,0.1),
              lty=2, col="darkgray") +
  facet_wrap(~ agegp, nrow=2, scales="free",
             labeller=as_labeller(lab.age)) +
  xlab("number of controls") + ylab("number of cases") +
  scale_shape_discrete(name="alcohol (g/day)",
                       labels=lab.alc) +
  scale_color_manual(values=col.tob,
                     name="tobacco (g/day)",
                     labels=lab.tob) +
  theme_bw()
```

ggplot2 version



Stanford MEDICINE | Data Studio

ComplexHeatmap

Bioconductor package

bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html

- Installation instructions
- Links to excellent vignettes

Simulated data

- 50 samples
- For each sample:
  - Clinical parameters in data frame **samples**
    - Sex
    - Age
    - Severity of symptoms: none, moderate, severe
  - Biological measurements in matrix **mtx**
    - 10 of type X
    - 15 of type Y
    - Some measurements depend upon one or more clinical parameters

ComplexHeatmap code: definitions for colors and labels

```
col.sym <- c(none="gold",
             moderate="olivedrab",
             severe="firebrick2")
col.sex <- c(M="skyblue", F="pink2")
grp.msr <- substr(rownames(mtx), 1, 1)
col.msr.lbl <- rep("black", nrow(mtx))
col.msr.lbl[ind.sym] <- col.sym["severe"]
fn1 <- function(str) paste("type", str)
```

ComplexHeatmap code: annotations

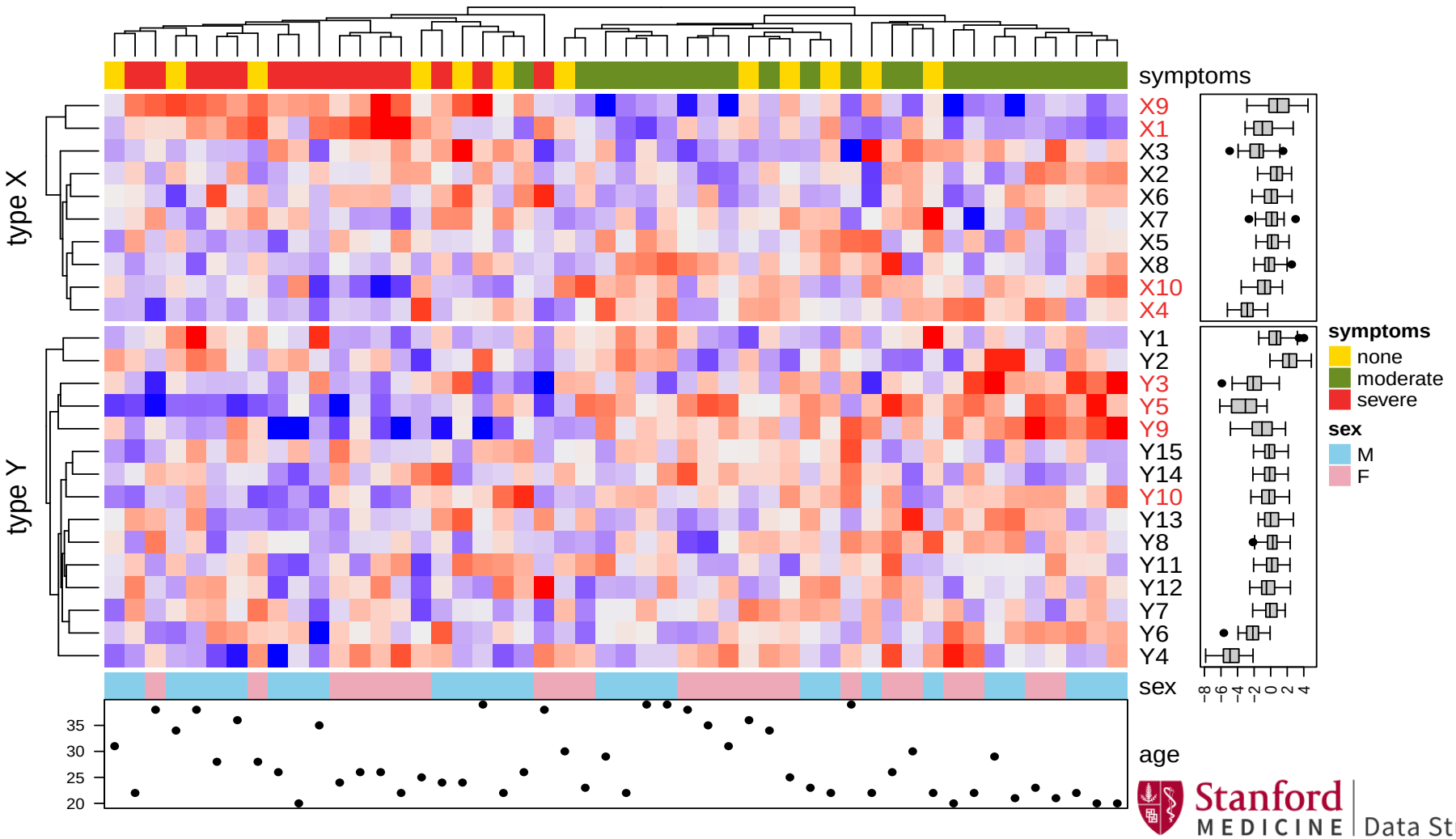
```
ha.top <- HeatmapAnnotation(df=samples[,"symptoms",drop=F],
                            col=list(symptoms=col.sym),
                            show_annotation_name=T)
ha.btm <- HeatmapAnnotation(df=samples[,"sex",drop=F],
                            col=list(sex=col.sex),
                            age=anno_points(samples$age, axis=
                            annotation_height=unit(c(0.5,2), "
                            show_annotation_name=T)
ha.row <- rowAnnotation(box=row_anno_boxplot(mtx, axis=T),
                        width=unit(2, "cm"))
```

ComplexHeatmap code: main heatmap

```
hmc <- Heatmap(mtx - rowMeans(mtx),
               name="hm",
               split=grp.msr,
               combined_name_fun=fn1,
               top_annotation=ha.top,
               bottom_annotation=ha.btm,
               row_names_gp=gpar(col=col.msr.lbl),
               show_column_names=F,
               show_heatmap_legend=F)
```

ComplexHeatmap plot

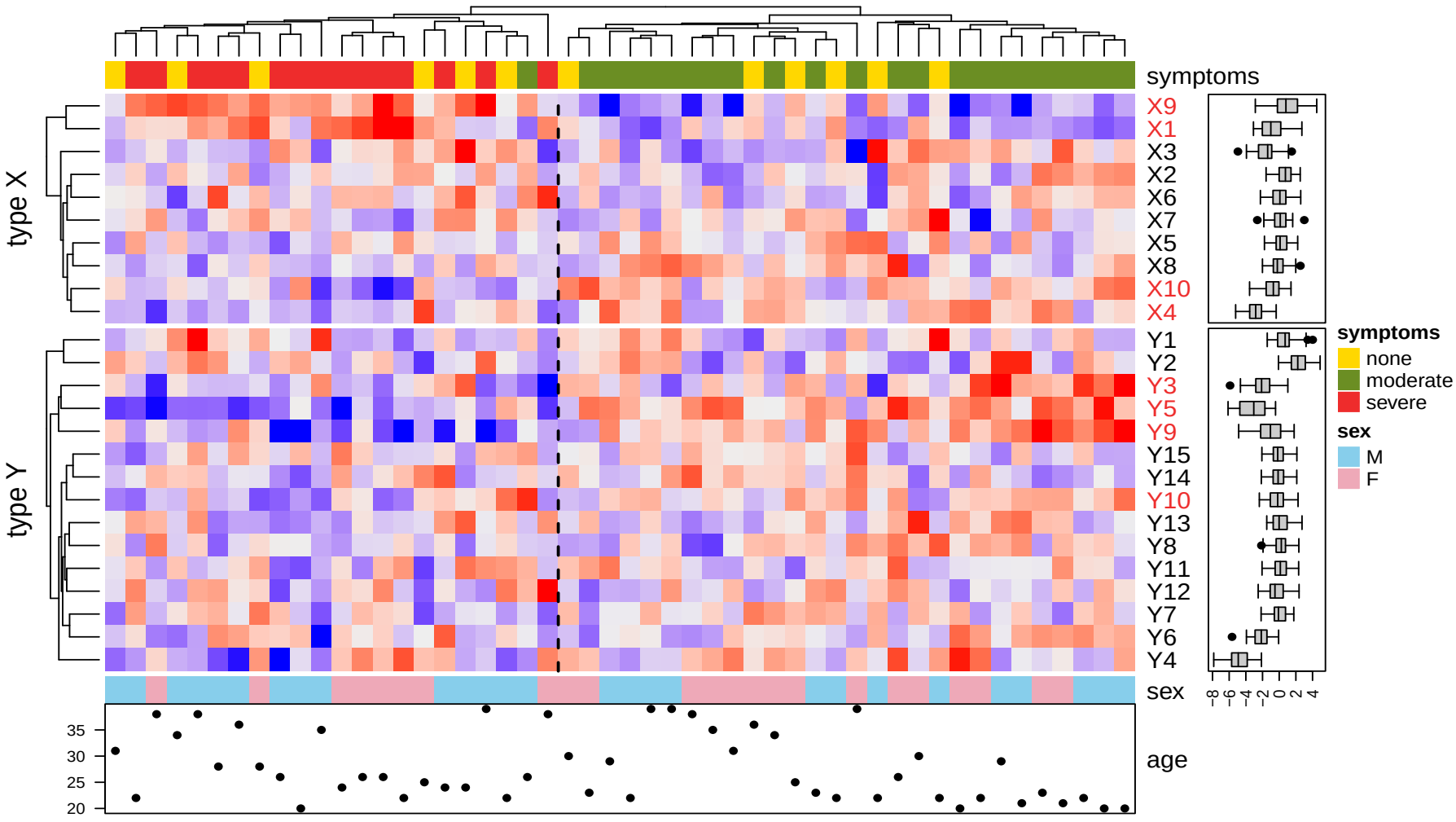
```
draw(hmc + ha.row)
```



And yet more: decorations

```
draw(hmc + ha.row)
for (nnn in 1:2) {
  decorate_heatmap_body("hm",
    {
      tree <- column_dend(hmc)
      ind <- cutree(as.hclust(tree), k=2)[order.dendrogram(tre
      n1 <- sum(ind == 1)
      x <- n1/length(ind)
      grid.lines(rep(x,2), y=c(0,1), gp=gpar(lty=2, lwd=2))
    },
  slice=nnn)
}
```

ComplexHeatmap plot with decorations



Stanford MEDICINE | Data Studio

Wrap-up

Choosing between **lattice** and **ggplot2**

- **lattice** paradigm is more like standard R, so easier to get started
- Once you develop an ear for the **ggplot2** dialect, it might be easier to guess how to do something new
- Easier to do moderate levels of customization with **ggplot2**
- **lattice** allows almost total control
- Customizing legends is painful

Finding help with **lattice** or **ggplot2**

- Lots of tutorials on Web
- R's online documentation in your installation
  - For example, **help(geom_line)** if you know command you need
  - **help(lattice)** has an overview and lists of functions
  - **help.start()** and go to search engine. Examples:
    - "panel" for **lattice**
    - "geom_" for **ggplot2**
- The Web
  - Including "R" in Google search not satisfactory. Instead:
    - Start search with "r-project" *or*
    - Use https://rseek.org
  - If you want to change default behavior in some way, somebody else probably has already done it.

ComplexHeatmap

- Can have no heatmap and just show annotation
- Can have multiple heatmaps side-by-side
- Can use additional heatmaps as annotation
- Excellent vignettes
  - Available from Bioconductor page
  - Included with installation
- Might want to try **superheat** package
  - Annotation on top or right
  - Only one annotation per side