Graduate School Class Reminders

- Maintain six feet of distancing
- Please sit in the same chair each class time
- Observe entry/exit doors as marked
- Use hand sanitizer when you enter/exit the classroom
- Use a disinfectant wipe/spray to wipe down your learning space before and after class
- Media Services: 414 955-4357 option 2

Documentation on the web

- CRAN: `http://cran.r-project.org`
- R manuals: `https://cran.r-project.org/manuals.html`
- SAS: `http://support.sas.com/documentation`
- Step-by-Step Programming with Base SAS 9.4 (SbS): `https://documentation.sas.com/api/docsets/basess/9.4/content/basess.pdf`
- SAS 9.4 Programmer s Guide: Essentials (PGE): `https://documentation.sas.com/api/docsets/lepg/9.4/content/lepg.pdf`
- Wiki: `https://wiki.biostat.mcw.edu` (MCW/VPN)
- `lattice`: `http://lmdvr.r-forge.r-project.org/figures/figures.html`

Graphical exploration with R: two-headed monster

- There are two graphics packages for R that have a similar syntax which makes learning both of them convenient
- The `graphics` package for routine daily usage is one of the *base* packages as described in Section 12
- Easy to use, it can create high-quality graphical figures, but there are limits to their customizability
- You can see the documentation with `library(help=graphics)`
- Depends on the `grDevices` package which has support for devices, colors, fonts, etc.

Graphical exploration with R: two-headed monster

- The `lattice` package which is one of the *recommended* packages can create more appealing customizable graphical figures of publication and grant proposal quality
- It builds upon `grDevices` and `graphics`
- With some added capabilities in `latticeExtra`
- You can see the documentation with `library(help=lattice)`
- But, the `lattice` package documentation is voluminous and it can take quite a while to read and understand
- Yet, the examples are very helpful; there are extra examples online that will dispell some of the mystery `http://lmdvr.r-forge.r-project.org/figures/figures.html`

Graphical exploration with R: two-headed monster

- DO NOT USE `ggplot2`!
- It is unacceptable for the following reasons
- The graphs are terrible: a gray background by default?
- `ggplot2` package doesn't come with R
- Its syntax doesn't resemble `graphics/lattice`
- It has no 3D capabilities

Graphical exploration with R: two-headed monster

- Common types of plots and their high-level functions

Type of plot	Dimensions	`graphics`	`lattice`
Box and whisker	1	`boxplot`	`bwplot`
Histogram	1	`hist`	`histogram`
Density	1	`plot.density`	`densityplot`
Quantile-quantile	1	`qqnorm`	`qqmath`
QQ	2	`qqplot`	`qq`
Scatter	2	`plot`	`xyplot`
Contour	3	`contour`	`contourplot`
Heat map	3	`image`	`levelplot`
Surface	3	`persp`	`wireframe`

Graphical exploration with R: the `graphics` package

- Let's explore the high-level `plot` function
- Many objects types have their own generic functions
  `> plot(object)` will call the function `plot.CLASS` where the `object` is of type `CLASS` like `plot.density`
- But, we are more concerned with the default: `plot.default`
- Here the arguments of primary interest that are shared/comparable between `graphics` and `lattice`

```
plot(x, y = NULL, col = 1,
     type = "p", cex = 1, pch = 1, lty = 1, lwd = 1,
     xlim = NULL, ylim = NULL,
     xlab = NULL, ylab = NULL)
```

Graphical exploration with R: the graphics package

```
plot(x, y = NULL, col = 1,
     type = "p", cex = 1, pch = 1, lty = 1, lwd = 1,
     xlim = NULL, ylim = NULL,
     xlab = NULL, ylab = NULL)
```

- x is a vector for the x-axis
- y same thing for the y-axis
  if y is not specified, then x is plotted as a time series
- common type settings: "p" for points, "l" for lines,
  "b" for both points and lines, "h" for vertical lines,
  "s" for stair steps/survival functions and
  "n" does not produce anything
- cex for the size of points and pch for the kind
- specify the line type by lty and the line width by lwd
- xlim is a length 2 vector for the limits of the x-axis
- xlab is the label for the x-axis: either a character string
  or an expression see `> ?plotmath`
- ylim/ylab same thing for the y-axis

Graphical exploration with R: the `graphics` package

```
plot(x, y = NULL, col = 1,
     type = "p", cex = 1, pch = 1, lty = 1, lwd = 1,
     xlim = NULL, ylim = NULL,
     xlab = NULL, ylab = NULL)
```

- `col` is the color to be used for the plot points/lines/etc.
- A color's name by character string: see `colors` for a list
- Or it can be an integer *marker* number
- This is based on the current color palette which can be customized by the `palette` function: default markers
- Choosing good colors can be tricky: stick to the powers of 2

$0 = 2^{-\infty}$	*white*	$1 = 2^0$	*black*	$2 = 2^1$	*red*
3	green	$4 = 2^2$	*blue*	5	cyan
6	magenta	7	yellow	$8 = 2^3$	*gray*

- If you need more, then see `bass/colors.R`

Graphical exploration with R: the `graphics` package

- Interactively, `plot` will create a graphics window with a panel
- Or multiple `plot` calls yielding multiple panels via
  `par(mfcol=c(rows, cols))` or
  `par(mfrow=c(rows, cols))`
  but often with very poor quality: see `lecture5.R`
- Low-level functions that overlay graphics on this panel follow
- Those that are self-explanatory and similar to `lattice`
- `lines(x, y=NULL, col=1, lty=1, lwd=1)`
- `points(x, y=NULL, col=1, pch=1, cex=1)`
- `text(x, y=NULL, labels, col=1, cex=1, pos=NULL)`
  pos 3
  2 (x, y) 4 positions go clock-wise starting at 6 o'clock
  1
- `legend` is NOT similar to `lattice`
- `legend(x, y=NULL, col=1, legend, lty=1, lwd=1, pch=1, cex=1, horiz=FALSE)`

Graphical exploration with R: the `graphics` package

- This is the same for the `lattice` package
- `> ?Devices` lists the graphical file formats supported
- We are going to restrict our attention to Adobe Portable Document Format (PDF)
- In an interactive R session, you can capture what is in the graphics window at any time by (in batch, this is an error)
  `> dev.copy2pdf(file="FILENAME.pdf")`
- In an interactive or batch R session, you can create a graphics file without a graphics window
- ```
  pdf(file="FILENAME.pdf")
  PLOTTING STATEMENTS
  dev.off()
  ```
- The latter can create graphics files with multiple pages each `plot` creates a page (except for multiple panels where each of these is a page)
  ```

Graphical exploration with R: the `lattice` package

```
xyplot(formula, data, groups = NULL, col = 1,
     type = "p", cex = 1, pch = 1, lty = 1, lwd = 1,
     xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,
     strip = strip.custom(strip.names = FALSE),
     layout = c(cols, rows), as.table = FALSE)
```

- `formula` of the form `y~x` for a single panel
  or `y~x|g1*g2*...` for panels of plots conditioned on the variables `g1*g2*...` which are typically factors
  if not factors, then specify `factor(g1)*factor(g2)*...`
- `data` is a data frame with `x` and `y`, etc.
- `groups=z` overlays data within the same panel based on different values of the variable `z`
- `layout` controls how multiple panels are configured, if any
- `as.table` controls the order multiple plans are drawn
  `as.table = TRUE` seems more logical than the default
- `strip` controls how the panels are labeled
  `strip.names = TRUE` seems more logical than the default

Graphical exploration with R: the `lattice` package

- Interactively, `xyplot` will create a graphics window
- Low-level functions can overlay graphics on this panel
- Those that are similar to `graphics` follow
  typically, just prefix an `l` on their names for lattice
- `llines(x, y=NULL, col=1, lty=1, lwd=1)` adds lines
- `lpoints(x, y=NULL, col=1, pch=1, cex=1)` adds points
- `ltext(x, y=NULL, col=1, labels, cex=1,`
  `pos=NULL, offset=0.5)` adds text
- And there is an additional one like `plot` itself
  `lplot.xy(list(x=x, y=y), col=1, type="p",`
  `cex=1, pch=1, lty=1, lwd=1)`

```
xyplot(formula, data, ## the easy way
panel=function(...){ panel.abline(h=0);panel.xyplot(...)})
xyplot(formula, data) ## or the hard way
update(trellis.last.object(),
       panel=function(...) {
           i=panel.number() ## call panel.FUNCTIONS and
           lFUNCTIONS })
```

Graphical exploration with R: the `lattice` package

- As we saw, there are three 3D functions in `graphics`: `contour`, `image` and `persp`
- Each of these functions has an interface like `NAME(x, y, Z)` where `x` and `y` are vectors and `Z` is a matrix
- This is a confusing interface to say the least
  i.e., how does `Z` relate to `x` and `y`?
- `lattice` has a more intuitive interface for the corresponding functions: `contourplot`, `levelplot` and `wireframe` as follows (with arguments analogous to `xyplot`)
- `NAME(z~x*y|g1*g2*..., data)` where `x`, `y` and `z` are all vectors in the data frame so their relationship is obvious
- Many of the same arguments as `xyplot`
- `groups` is a notable exception

Graphical exploration with R: the `lattice` package

- There is a `legend` argument but it seems difficult to use
  see `key` instead
- `key=list(border="black", x=x, y=y,`
  `text=list(...), ..., lines=list(...))`
  or `..., points=list(...))`
  where `x` and `y` in [0, 1] of the displayed area
- `xlim` and `ylim` are too simplistic for many needs
  `scales` is more flexible
- ```
  scales=list(
  x=list(limits=c(low, high), log=TRUE),
  y=list(at=c(1, 2, 4, 8),
    labels=c("a", "b", "c", "d")))
  ```
- `at` are the tick marks drawn: can be increasing or decreasing
- `labels` are the values to display there
  ```

Graphical exploration with R: the `lattice` package

- `trellis.par.get()` to show default settings
- and `trellis.par.set()` to alter them
- for example, slide 13 shows the default for `xyplot` is `col = 1` however, that is NOT right: more complicated than that

```
trellis.par.get("plot.symbol")
a$col = 1
trellis.par.set("plot.symbol", a)
```

Graphical exploration with R: the `latticeExtra` package

- You can overlay two or more `trellis` objects
- For example, `doubleYScale` to create a second y-axis
- Combine two objects with the `as.layer` function
- Combine multiple objects with the `c` function