PRINCIPLES OF GRAPH CONSTRUCTION

Frank E Harrell Jr
Department of Biostatistics
Vanderbilt University School of Medicine
f.harrell@vanderbilt.edu
biostat.mc.vanderbilt.edu at Jump: StatGraphCourse
Office of Biostatistics, FDA CDER
frank.harrell@fda.hhs.gov
Blog: fharrell.com Twitter: @f2harrell

DIA/FDA STATISTICS FORUM
NORTH BETHESDA MD
2017-04-24

Chapter 1

Principles of Graph Construction

The ability to construct clear and informative graphs is related to the ability to understand the data. There are many excellent texts on statistical graphics (many of which are listed at the end of this chapter). Some of the best are Cleveland's 1994 book *The Elements of Graphing Data* and the books by Tufte. The suggestions for making good statistical graphics outlined here are heavily influenced by Cleveland's 1994 book. See also the excellent special issue of *Journal of Computational and Graphical Statistics* vol. 22, March 2013.

1.1 Graphical Perception

- Goals in communicating information: reader perception of data values and of data patterns. Both accuracy and speed are important.
- Pattern perception is done by

  detection : recognition of geometry encoding physical values

  assembly : grouping of detected symbol elements; discerning overall patterns in data

  estimation : assessment of relative magnitudes of two physical values
- For estimation, many graphics involve discrimination, ranking, and estimation of ratios
- Humans are not good at estimating differences without directly seeing differences (especially for steep curves)
- Humans do not naturally order color hues
- Only a limited number of hues can be discriminated in one graphic
- Weber's law: The probability of a human detecting a difference in two lines is related to the ratio of the two line lengths

- This is why grid lines and frames improve perception and is related to the benefits of having multiple graphs on a common scale.
  - eye can see ratios of filled or of unfilled areas, whichever is most extreme
- For categorical displays, sorting categories by order of values attached to categories can improve accuracy of perception. Watch out for over-interpretation of extremes though.
- The aspect ratio (height/width) does not have to be unity. Using an aspect ratio such that the average absolute curve angle is 45° results in better perception of shapes and differences (banking to 45°).
- Optical illusions can be caused by:
  - hues, e.g., red is emotional. A red area may be perceived as larger.
  - shading; larger regions appear to be darker
  - orientation of pie chart with respect to the horizon
- Humans are bad at perceiving relative angles (the principal perception task used in a pie chart)
- Here is a hierarchy of human graphical perception abilities:

  1. Position along a common scale (most accurate task)

2. Position along identical nonaligned scales
3. Length
4. Angle and slope
5. Area
6. Volume
7. Color: hue (red, green, blue, etc.), saturation (pale/deep) and lightness
   - Hue can give good discrimination but poor ordering

1.2 General Suggestions

- Exclude unneeded dimensions (e.g. width, depth of bars)
- "Make the data stand out. Avoid Superfluity"; Decrease ink to information ratio
- "There are some who argue that a graph is a success only if the important information in the data can be seen in a few seconds. . . . Many useful graphs require careful, detailed study."
- When actual data points need to be shown and they are too numerous, consider showing a random sample of the data.

- Omit “chartjunk”
- Keep continuous variables continuous; avoid grouping them into intervals. Grouping may be necessary for some tables but not for graphs.
- Beware of subsetting the data finer than the sample size can support; conditioning on many variables simultaneously (instead of multivariable modeling) can result in very imprecise estimates

Murrell [23] has an excellent summary of recommendations:

- Display data values using position or length.
- Use horizontal lengths in preference to vertical lengths.
- Watch your data–ink ratio.
- Think very carefully before using color to represent data values.
- Do not use areas to represent data values.

- *Please* do not use angles or slopes to represent data values.

- *Please, please* do not use volumes to represent data values.

1.3 Tufte on "Chartjunk"

> Chartjunk does not achieve the goals of its propagators. The overwhelming fact of data graphics is that they stand or fall on their content, gracefully displayed. Graphics do not become attractive and interesting through the addition of ornamental hatching and false perspective to a few bars. Chartjunk can turn bores into disasters, but it can never rescue a thin data set. The best designs . . . are *intriguing and curiosity-provoking*, drawing the viewer into the wonder of the data, sometimes by narrative power, sometimes by immense detail, and sometimes by elegant presentation of simple but interesting data. But no information, no sense of discovery, no wonder, no substance is generated by chartjunk.

— Tufte p. 121, 1983

1.4 Tufte's Views on Graphical Excellence

"Excellence in statistical graphics consists of complex ideas communicated with clarity, precision, and efficiency. Graphical displays should

- show the data
- induce the viewer to think about the substance rather than about methodology, graphic design, the technology of graphic production, or something else
- avoid distorting what the data have to say
- present many numbers in a small space
- make large data sets coherent
- encourage the eye to compare different pieces of data
- reveal the data at several levels of detail, from a broad overview to the fine structure
- serve a reasonably clear purpose: description, exploration, tabulation, or decoration
- be closely integrated with the statistical and verbal descriptions of a data set."

1.5 Formatting

- Tick Marks should point outward
- x- and y-axes should intersect to the left of the lowest x value and below the lowest y value, to keep values from being hidden by axes
- Minimize the use of remote legends. Curves can be labeled at points of maximum separation (see the Hmisc `labcurve` function).

1.6 Color, Symbols, and Line Styles

- Some symbols (especially letters and solids) can be hard to discern
- Use hues if needed to add another dimension of information, but try not to exceed 3 different hues. Instead, use different saturations in each of the three different hues.
- Make notations and symbols in the plots as consistent as possible with other parts, like tables and texts
- Different dashing patterns are hard to read especially when curves inter-twine or when step functions are being displayed

- An effective coding scheme for two lines is to use a thin black line and a thick gray scale line

1.7 Scaling

- Consider the inclusion of 0 in your axis. Many times it is essential to include 0 to tell the full story. Often the inclusion of zero is unnecessary.
- Use a log scale when it is important to understand percent change of multiplicative factors or to cure skewness toward large values
- Humans have difficulty judging steep slopes; bank to $45°$, i.e., choose the aspect ratio so that average absolute angle in curves is $45°$.

1.8 Displaying Estimates Stratified by Categories

- Perception of relative lengths is most accurate — areas of pie slices are difficult to discern
- Bar charts have many problems:
  - High ink to information ratio
  - Error bars cause perception errors

  - Can only show one-sided confidence intervals well
  - Thick bars reduce the number of categories that can be shown
  - Labels on vertical bar charts are difficult to read
- Dot plots are almost always better
- Consider multi-panel side-by-side displays for comparing several contrasting or similar cases. Make sure the scales in both x and y axes are the same across different panels.
- Consider ordering categories by values represented, for more accurate perception

1.9 Displaying Distribution Characteristics

- When only summary or representative values are shown, try to show their confidence bounds or distributional properties, e.g., error bars for confidence bounds or box plot
- It is better to show confidence limits than to show ± 1 standard error
- Often it is better still to show variability of *raw* values (quartiles as in a box plot so as to not assume normality, or S.D.)

- For a quick comparison of distributions of a continuous variable against many categories, try box plots.
- When comparing two or three groups, overlaid empirical distribution function plots may be best, as these show all aspects of the distribution of a continuous variable.

1.10 Showing Differences

- Often the only way to perceive differences accurately is to actually compute differences; then plot them
- It is not a waste of space to show stratified estimates and differences between them on the same page using multiple panels
- This also addresses the problem that confidence limits for differences cannot be easily derived from intervals for individual estimates; differences can easily be significant even when individual confidence intervals overlap.
- Humans can't judge differences between steep curves; one needs to actually compute differences and plot them.

The plot in figure 1.1 shows confidence limits for indi-

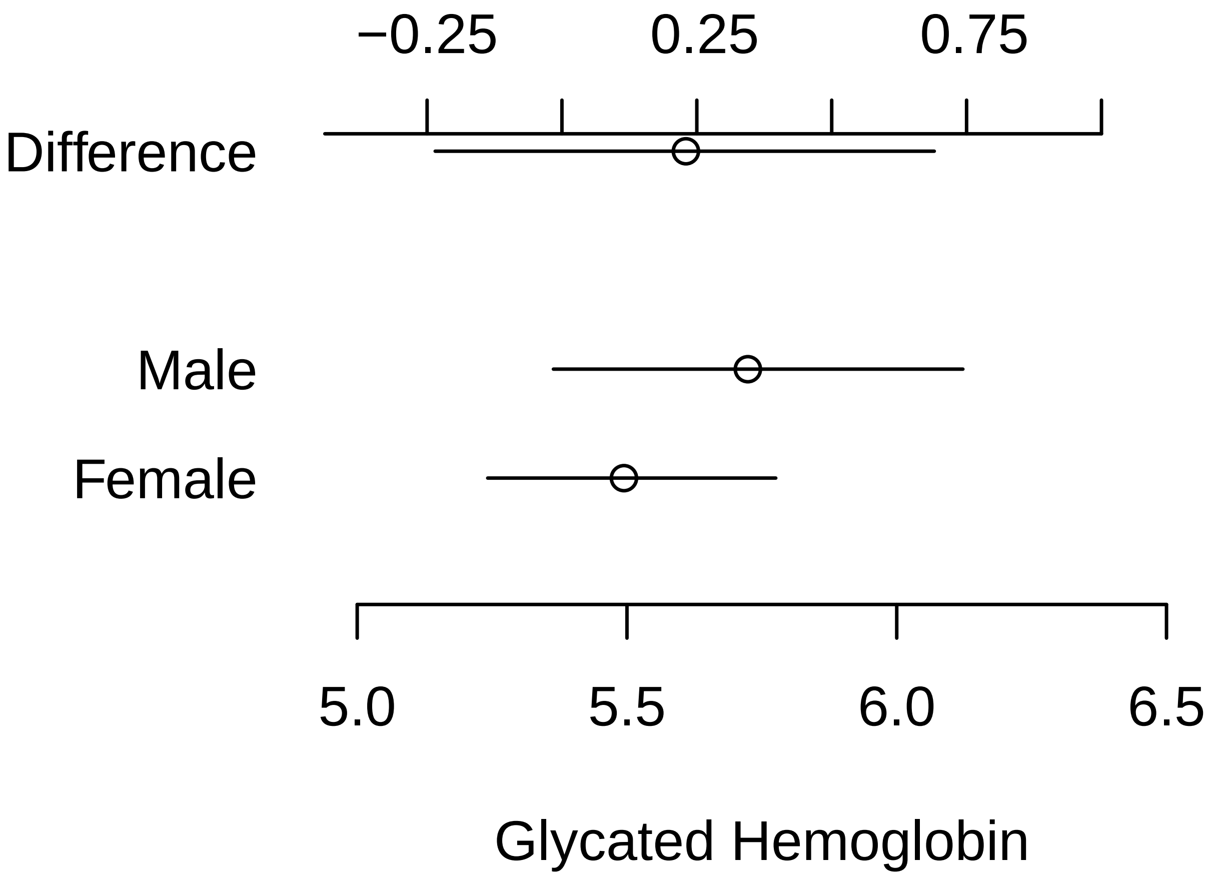


Figure 1.1: *Means and nonparametric bootstrap 0.95 confidence limits for glycated hemoglobin for males and females, and confidence limits for males - females. Lower and upper x-axis scales have same spacings but different centers. Confidence intervals for differences are generally wider than those for the individual constituent variables.*

vidual means, using the nonparametric bootstrap percentile method, along with bootstrap confidence intervals for the difference in the two means. The R code used to produce this figure is below.

```
attach(diabetes)
bootmean ← function(x,B=1000) {
  w ← smean.cl.boot(x, B=B, reps=T)
  reps ← attr(w,'reps')
  attr(w,'reps') ← NULL
  list(stats=w,reps=reps)
}

set.seed(1)
male   ← bootmean(glyhb[gender=='male'])
female ← bootmean(glyhb[gender=='female'])
dif ← c(mean=male$stats['Mean']-female$stats['Mean'],
         quantile(male$reps-female$reps, c(.025,.975)))
male   ← male$stats
female ← female$stats

par(mar=c(4,6,4,1))
plot(0,0,xlab='Glycated Hemoglobin',ylab='',
     xlim=c(5,6.5),ylim=c(0,4), axes=F)
axis(1)
axis(2, at=c(1,2,4),
     labels=c('Female','Male','Difference'),
     las=1, adj=1, lwd=0)


points(c(male[1],female[1]), 2:1)
segments(female[2], 1, female[3], 1)
segments(male[2],   2,   male[3], 2)

offset ← mean(c(male[1],female[1])) - dif[1]
points(dif[1] + offset, 4)
segments(dif[2]+offset, 4, dif[3]+offset, 4)

at ← c(-.5,-.25,0,.25,.5,.75,1)
axis(3, at=at+offset, label=format(at))
```

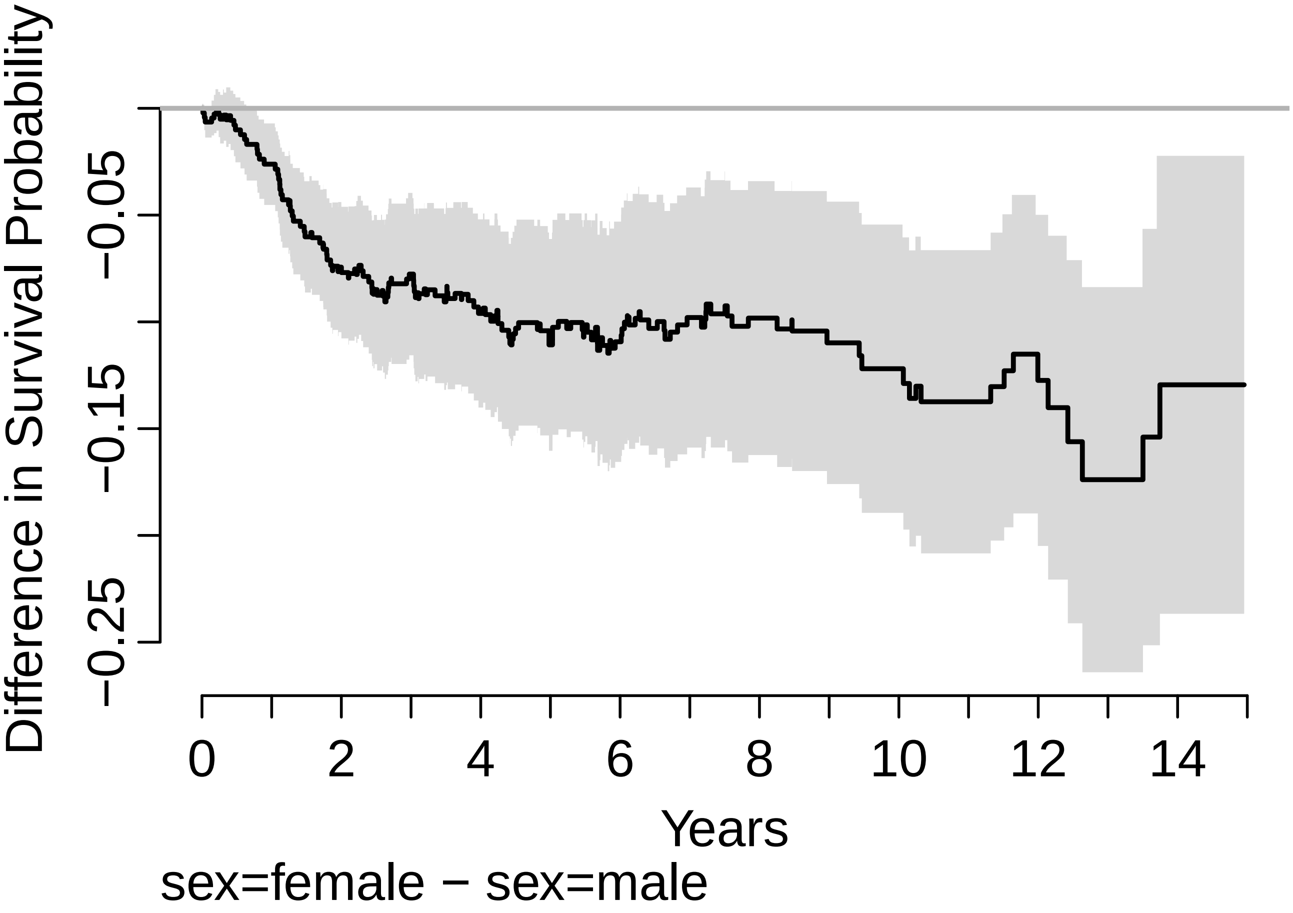


Figure 1.2: *Difference in two Kaplan–Meier survival curve estimates with pointwise 0.95 confidence bands for the difference; produced by the* `survplotdiff` *function in the* `rms` *package.*

1.11 Displaying Uncertainty

There are are least five ways of depicting the uncertainty of statistical estimates in graphs:

1. Bayesian posterior densities
2. error bars showing confidence limits
3. confidence bands drawn using two lines
4. shaded confidence bands
5. continuously graduated shading

Examples of approaches 2-4 appear in later sections. Bayesian posterior distributions convey the most accurate perception of uncertainty, and are easy to construct for a scalar parameter such as a single group mean. Continuous shading as developed by Jackson (2008) has several advantages (especially when estimating a function evaluated at many points) relating to its provision of the correct psychological effect of the limitations of information. Jackson has developed an R package called `denstrip` implementing these ideas. The example graphic below from the right panel of his Figure 5 shows the beauty of this approach in conveying uncertainty about forecasts into the future.

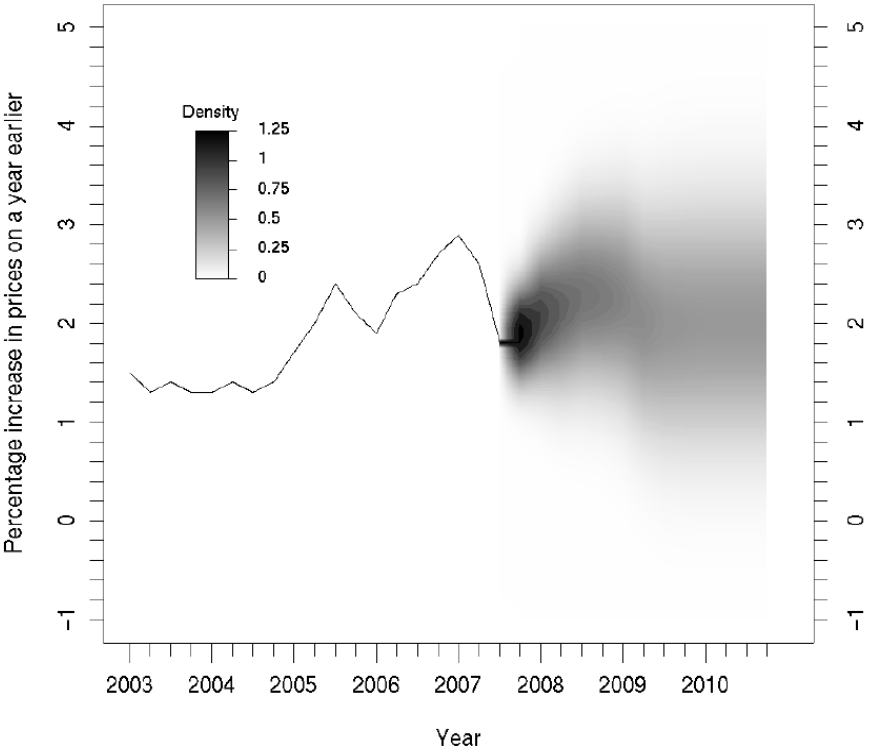


Figure 1.3: *Displaying uncertainty with shading*

1.12 Choosing the Best Graph Type

The recommendations that follow are good on the average, but be sure to think about alternatives for your particular data set. For nonparametric trend lines, it is advisable to add a “rug” plot to show the density of the data used to make the nonparametric regression estimate. Alternatively, use the bootstrap to derive nonparametric confidence bands for the nonparametric smoother.

1.12.1 Single Categorical Variable

Use a dot plot or horizontal bar chart to show the proportion corresponding to each category. Second choices for values are percentages and frequencies. The total sample size and number of missing values should be displayed somewhere on the page. If there are many categories and they are not naturally ordered, you may want to order them by the relative frequency to help the reader estimate values.

1.12.2 Single Continuous Numeric Variable

An empirical cumulative distribution function, optionally showing selected quantiles, conveys the most information and requires no grouping of the variable. A box plot will show selected quantiles effectively, and box plots are especially useful when stratifying by multiple categories of another variable. Histograms are also possible.

1.12.3 Categorical Response Variable vs. Categorical Ind. Var.

This is essentially a frequency table. It can also be depicted graphically

1.12.4 Categorical Response vs. a Continuous Ind. Var.

Choose one or more categories and use a nonparametric smoother to relate the independent variable to the proportion of subjects in the categories of interest. Show a rug plot on the x-axis.

1.12.5 Continuous Response Variable vs. Categorical Ind. Var.

If there are only two or three categories, superimposed empirical cumulative distribution plots with selected quantiles can be quite effective. Also consider box plots, or a dot plot with error bars, to depict the median and outer quartiles. Occasionally, a back-to-back histogram can be effective for two groups (see the Hmisc `histbackback` function).

1.12.6 Continuous Response vs. Continuous Ind. Var.

A nonparametric smoother is often ideal. You can add rug plots for the x- and y-axes, and if the sample size is not too large, plot the raw data. If you don't trust nonparametric smoothers, group the x-variable into intervals having a given number of observations, and for each x-interval plot characteristics (3 quartiles or mean $\pm$ 2 SD, for example) vs. the mean x in the interval. This is done automatically with the Hmisc `xYplot` function with the `methods='quantile'` option.

1.13 Conditioning Variables

You can condition (stratify) on one or more variables by making separate pages by strata, by making separate panels within a page, and by superposing groups of points (using different symbols or colors) or curves within a panel. The actual method of stratifying on the conditional variable(s) depends on the type of variables.

Categorical variable(s) : The only choice to make in conditioning (stratifying) on categorical variables is whether to combine any low-frequency categories. If you decide to combine them on the basis of relative frequencies you can use the `combine.levels` function in Hmisc.

Continuous numeric variable(s) : Unfortunately, to condition on a continuous variable without the use of a parametric statistical model, one must split the variable into intervals. The first choice is whether the intervals of the numeric variable should be overlapping or non-overlapping. For the former the built-in `equal.count` function can be used for a paneling or grouping variable in trellis graphics (these overlapping intervals are called “shingles” in trellis). For non-overlapping intervals the Hmisc `cut2` function is a good choice because of its many options and compact labeling.

1.14 Software

Recommended software for statistical graphics: R

- Base graphics functions (an excellent introduction for R is by Marc Schwartz at `http://cran.r-project.org/doc/Rnews/Rnews_2003-2.pdf`)

- R `lattice` packages for multi-panel displays and more (for R documentation with graphical output see `http://rgm3.lab.nig.ac.jp/RGM/R_image_list?package=lattice&init=true`; see also Deepayan (2008))

- R `ggplot2` package implementing much of Wilkinson (2005) (see `http://docs.ggplot2.org/current/`)

See R graphics galleries such as `http://scs.math.yorku.ca/index.php/R_Graphs_Gallery`, `http://www.sr.bham.ac.uk/~ajrs/R/r-gallery.html`, `http://rgraphgallery.blogspot.com`.

See `ctspedia.org` for much useful information about clinical trials and clinical safety graphics.

Bibliography

[1] C. F. Alzola and F. E. Harrell. *An Introduction to* S *and the* `Hmisc` *and* `Design` *Libraries*. Available from `http://biostat.mc.vanderbilt.edu/s/Hmisc`.

[2] O. Amit, R. M. Heiberger, and P. W. Lane. Graphical approaches to the analysis of safety data from clinical trials. *Pharmaceutical Statistics*, 7:20–35, 2008.

[3] F. J. Anscombe. Graphs in statistical analysis. *American Statistician*, 27:17–21, 1973.

[4] J. Bertin. *Graphics and Graphic Information-Processing*. de Gruyter, Berlin, 1981.

[5] D. B. Carr and S. M. Nusser. Converting tables to plots: A challenge from Iowa State. *Statistical Computing and Graphics Newsletter, ASA*, December 1995.

[6] C.-H. Chen, W. Härdle, and A. Unwin (eds.) *Handbook of Data Visualization*. Springer, New York, 2008.

[7] W. S. Cleveland. Graphs in scientific publications (c/r: 85v39 p238-239). *American Statistician*, 38:261–269, 1984.

[8] W. S. Cleveland. *Visualizing Data*. Hobart Press, Summit, NJ, 1993.

[9] W. S. Cleveland. *The Elements of Graphing Data*. Hobart Press, Summit, NJ, 1994.

[10] W. S. Cleveland and R. McGill. A color-caused optical illusion on a statistical graph. *American Statistician*, 37:101–105, 1983.

[11] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79:531–554, 1984.

[12] J. T. Connor. Statistical graphics in AJG: Save the ink for the information. *American Journal of Gastroenterology*, 104:1624–1630, 2009.

[13] S. Deepayan. Multivariate Data Visualization with R. Springer, New York, 2008.

[14] M. Friendly. The golden age of statistical graphics. *Statistical Science*, 23:502–535, 2008.

[15] A. Gelman, C. Pasarica, and R. Dodhia. Let's practice what we preach: Turning tables into graphs. *The American Statistician*, 56:121–130, 2002.

[16] F. E. Harrell. *Regression Modeling Strategies*. New York: Springer, 2001.

[17] F. E. Harrell. *Statistical Tables and Plots using S and LaTeX*. Available from http://biostat.mc.vanderbilt.edu/twiki/pub/Main/StatRepor

[18] G. T. Henry. *Graphing Data*. Sage, Newbury Park, CA, 1995.

[19] C. Jackson. Displaying uncertainty with shading. *The American Statistician*, 62:340–347, 2008.

[20] W. G. Jacoby. *Statistical Graphics for Univariate and Bivariate Data*. Thousand Oaks CA: SAGE Publications, 1997.

[21] X. Li, J. Buechner, P. Tarwater, and A. Muñoz. A diamond-shaped equiponderant graphical display of the effects of two categorical predictors on continuous outcomes. *The American Statistician*, 57:193–199, 2003.

[22] D. McNeil. On graphing paired data. *American Statistician*, 46:307–311, 1992.

[23] P. Murrell. Infovis and statistical graphics: Comment. *J Comp Graph Stat*, 22(1):33–37, 2013.

[24] S. M. Powsner and E. R. Tufte. Graphical summary of patient status. *Lancet*, 344:386–389, 1994.

[25] P. R. Rosenbaum. Exploratory plots for paired data. *American Statistician*, 43:108–109, 1989.

[26] P. D. Sasieni and P. Royston. Dotplots. *Applied Statistics*, 45:219–234, 1996.

[27] P. A. Singer and A. R. Feinstein. Graphical display of categorical data. *Journal of Clinical Epidemiology*, 46:231–236, 1993.

[28] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.

[29] E. R. Tufte. *Envisioning Information*. Graphics Press, Cheshire, Connecticut, 1990.

[30] E. R. Tufte. *Visual Explanations*. Graphics Press, Cheshire, CT, 1997.

[31] E. R. Tufte. *Beautiful Evidence*. Graphics Press, Cheshire, CT, 2006.

[32] H. Wainer. How to display data badly. *American Statistician*, 38:137, 1984.

[33] H. Wainer. Three graphic memorials. *Chance*, 7:52–55, 1994.

[34] H. Wainer. Depicting error. *American Statistician*, 50:101–111, 1996.

[35] H. Wainer. Improving data displays: Ours and the media's. *Chance*, 20:8-15, 2007.

[36] A. Wallgren, B. Wallgren, R. Persson, U. Jorner, and J. Haaland. *Graphing Statistics & Data*. Sage Publications, Thousand Oaks, 1996.

[37] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, San Francisco, 2004.

[38] L. Wilkinson. *The Grammar of Graphics, Second Edition*. Springer, New York, 2005.

Chapter 2

Examples

2.1 General Examples

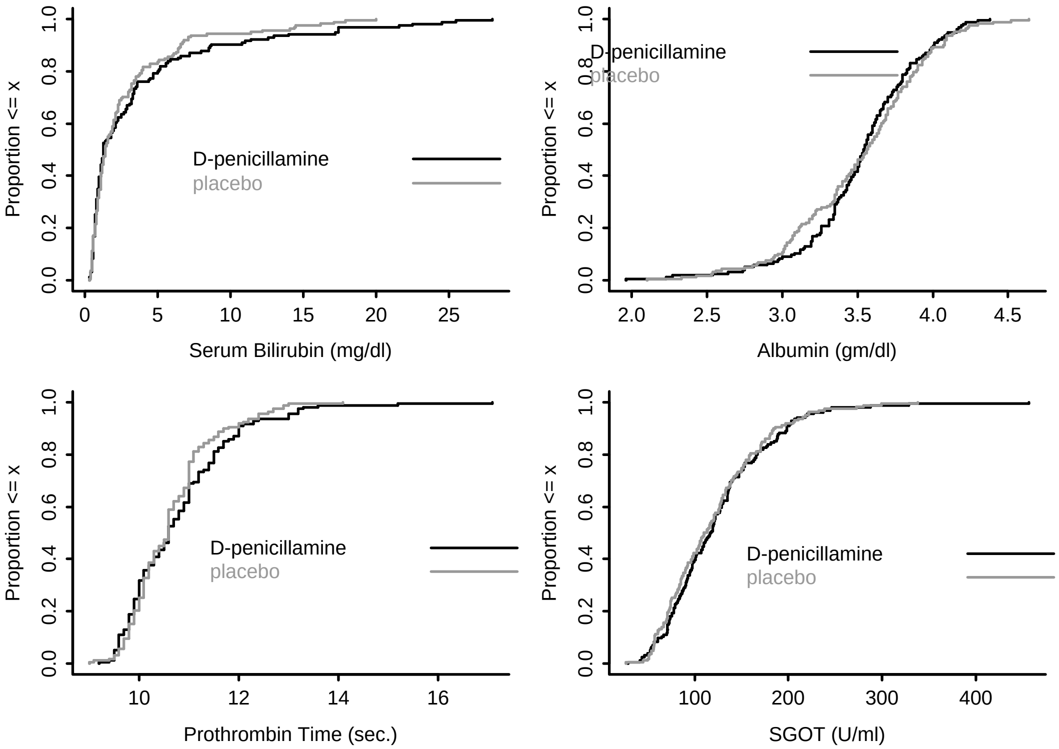


Figure 2.1: *Empirical cumulative distribution function for four continuous variables in the* `pbc` *dataset, stratified by randomized treatment*

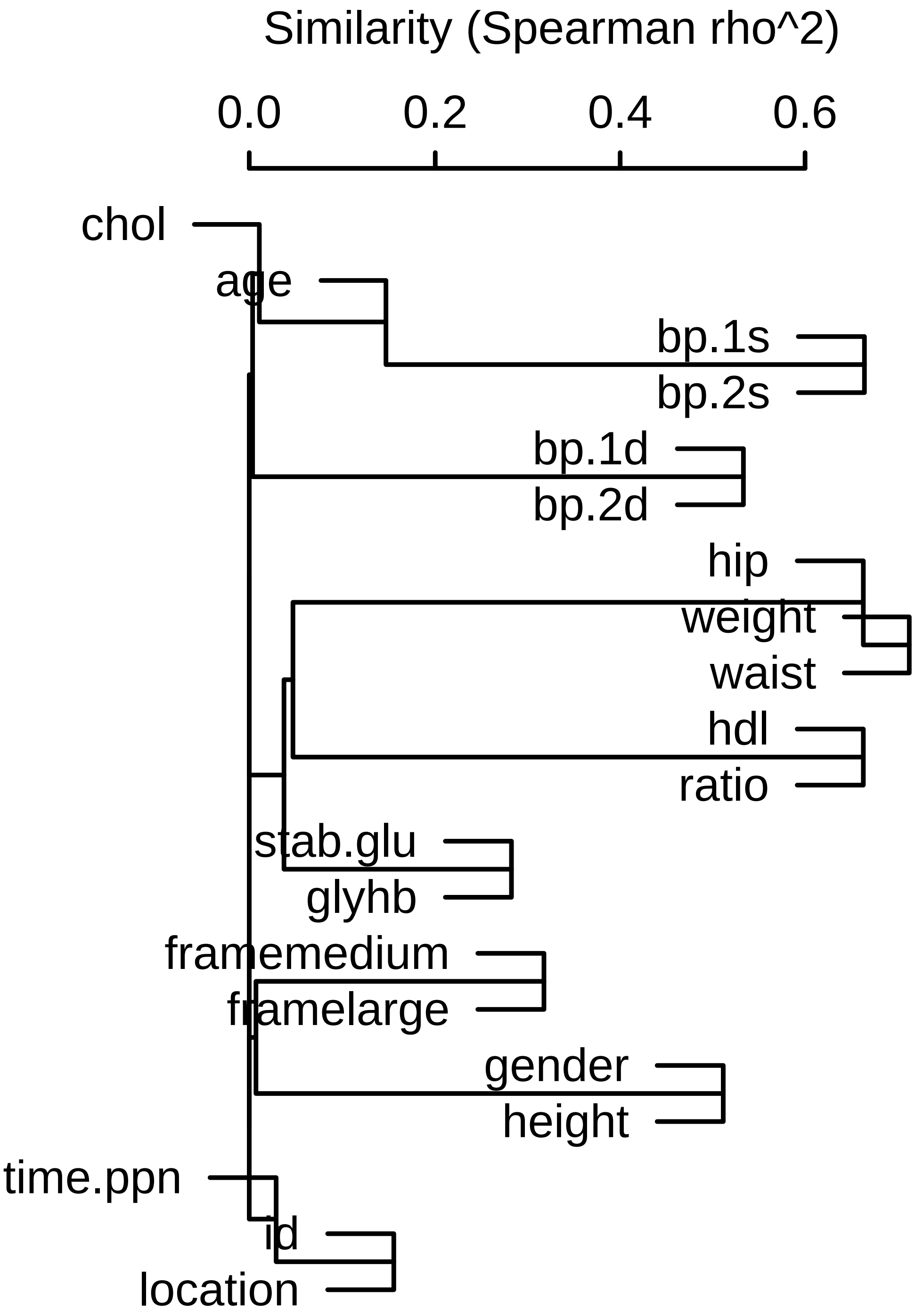


Figure 2.2: *Clusters of variables from `diabetes` using pairwise Spearman* ρ^2 *as the similarity measure*

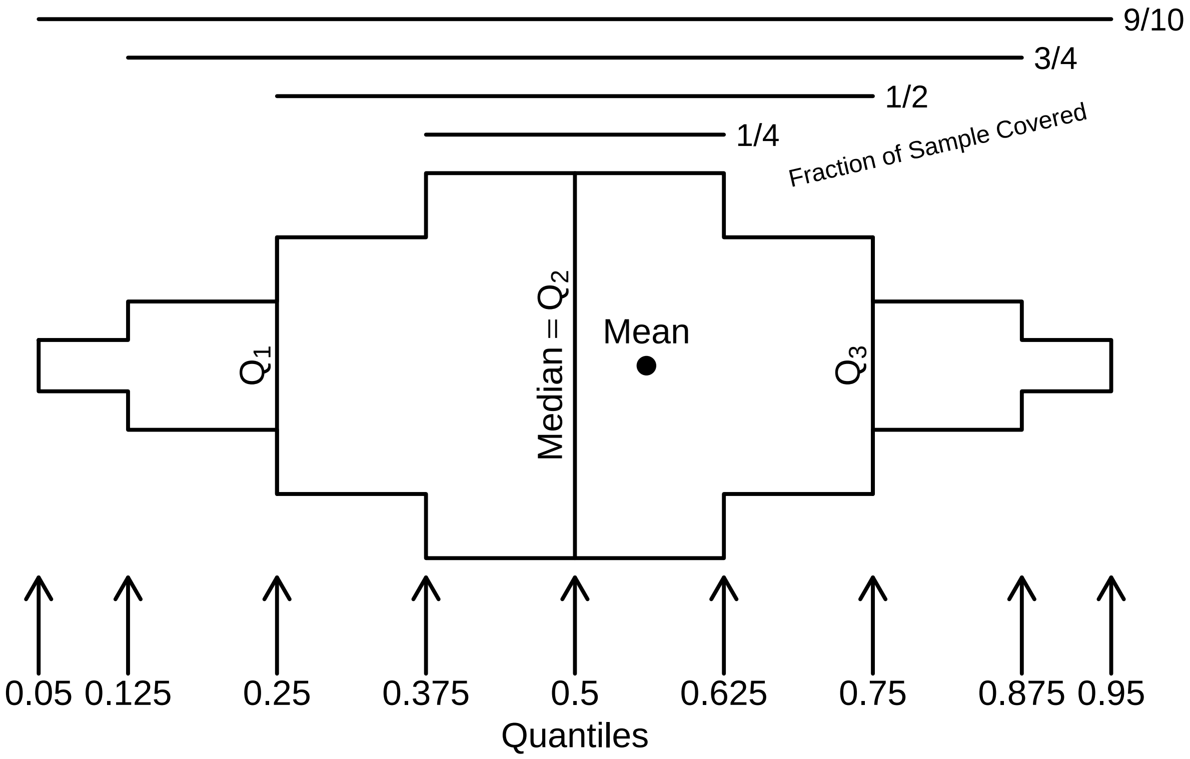


Figure 2.3: *Key for extended box plot*

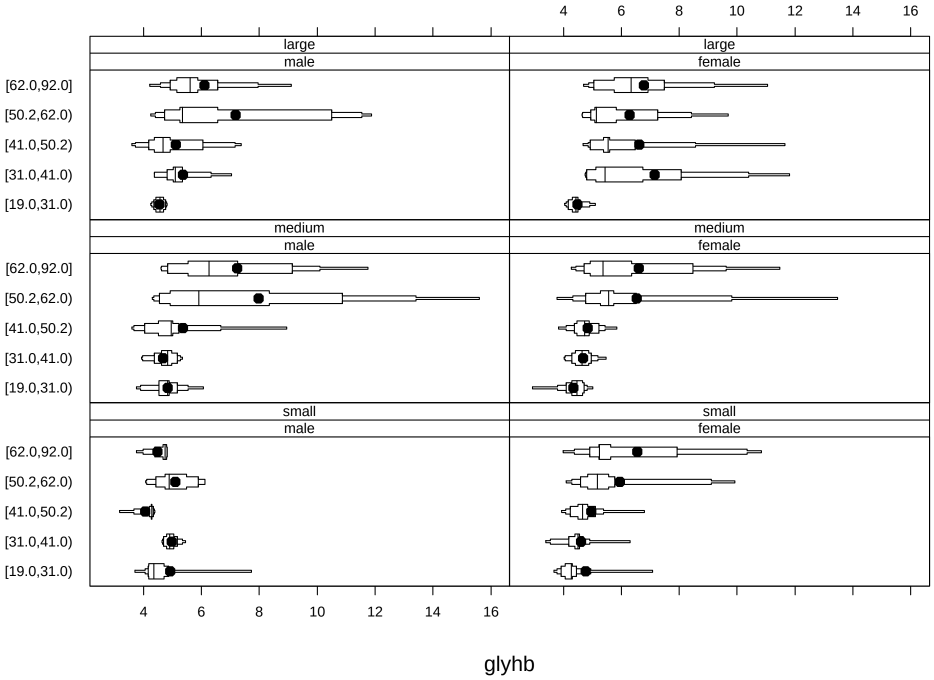


Figure 2.4: *Extended box plot displaying quantiles such that 0.25, 0.5, 0.75, and 0.9 of the data are contained within each pair of quantiles. The median is shown with a line, and the mean with a dot.*

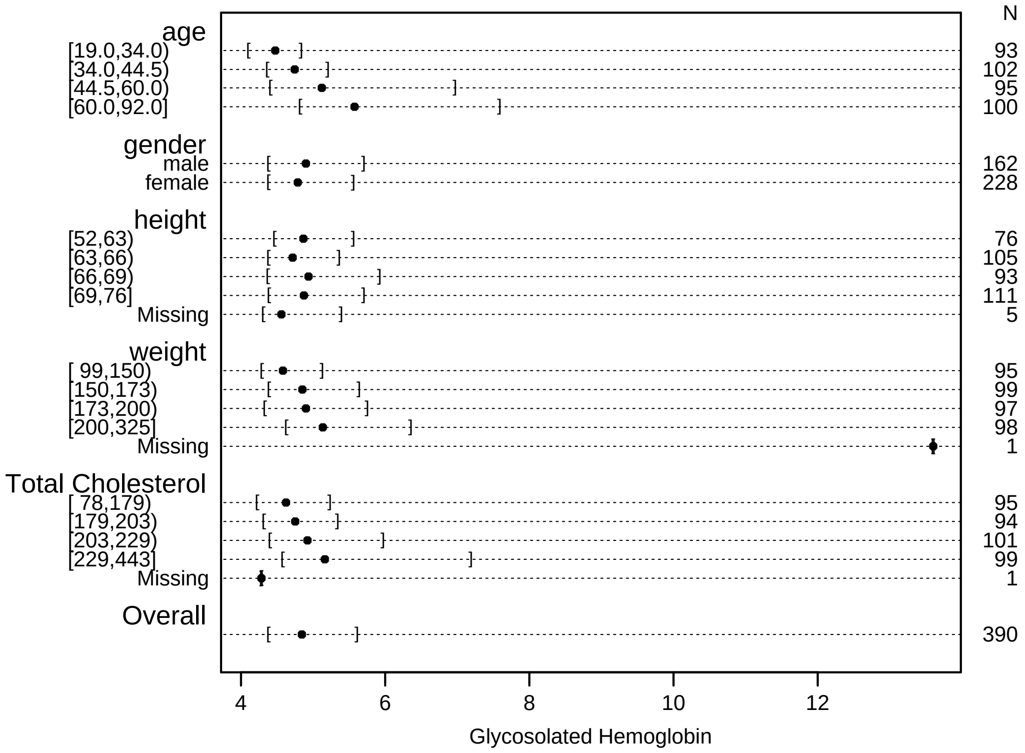


Figure 2.5: *Quartiles of `glyhb` stratified separately by several variables*

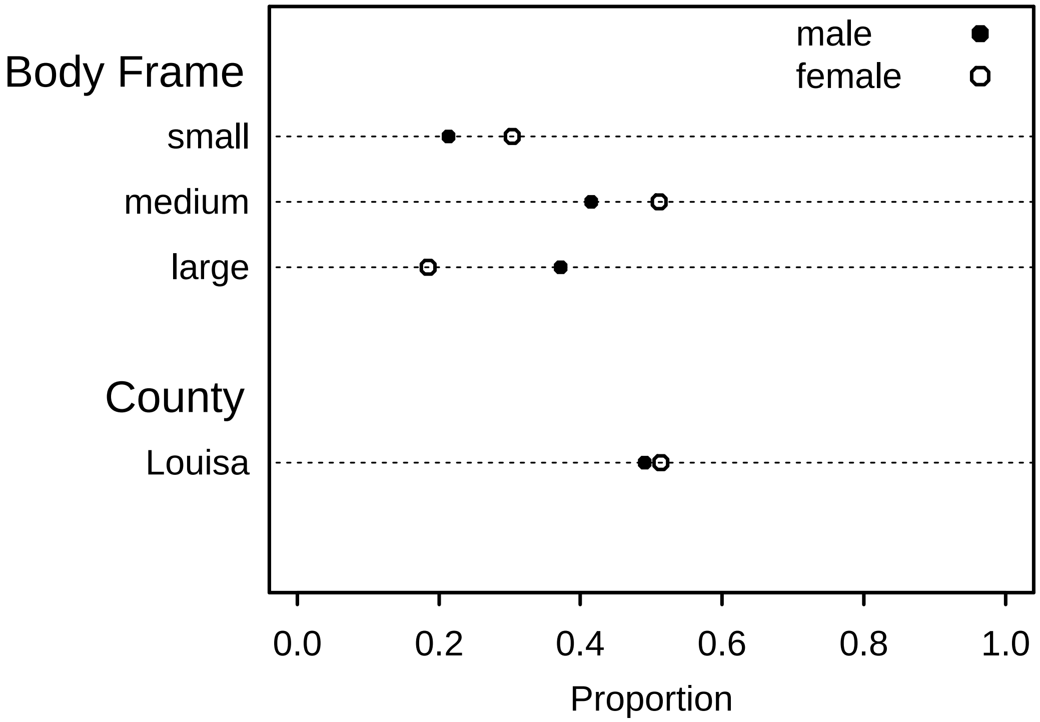


Figure 2.6: *Distribution of categorical varibles stratified by `gender`*

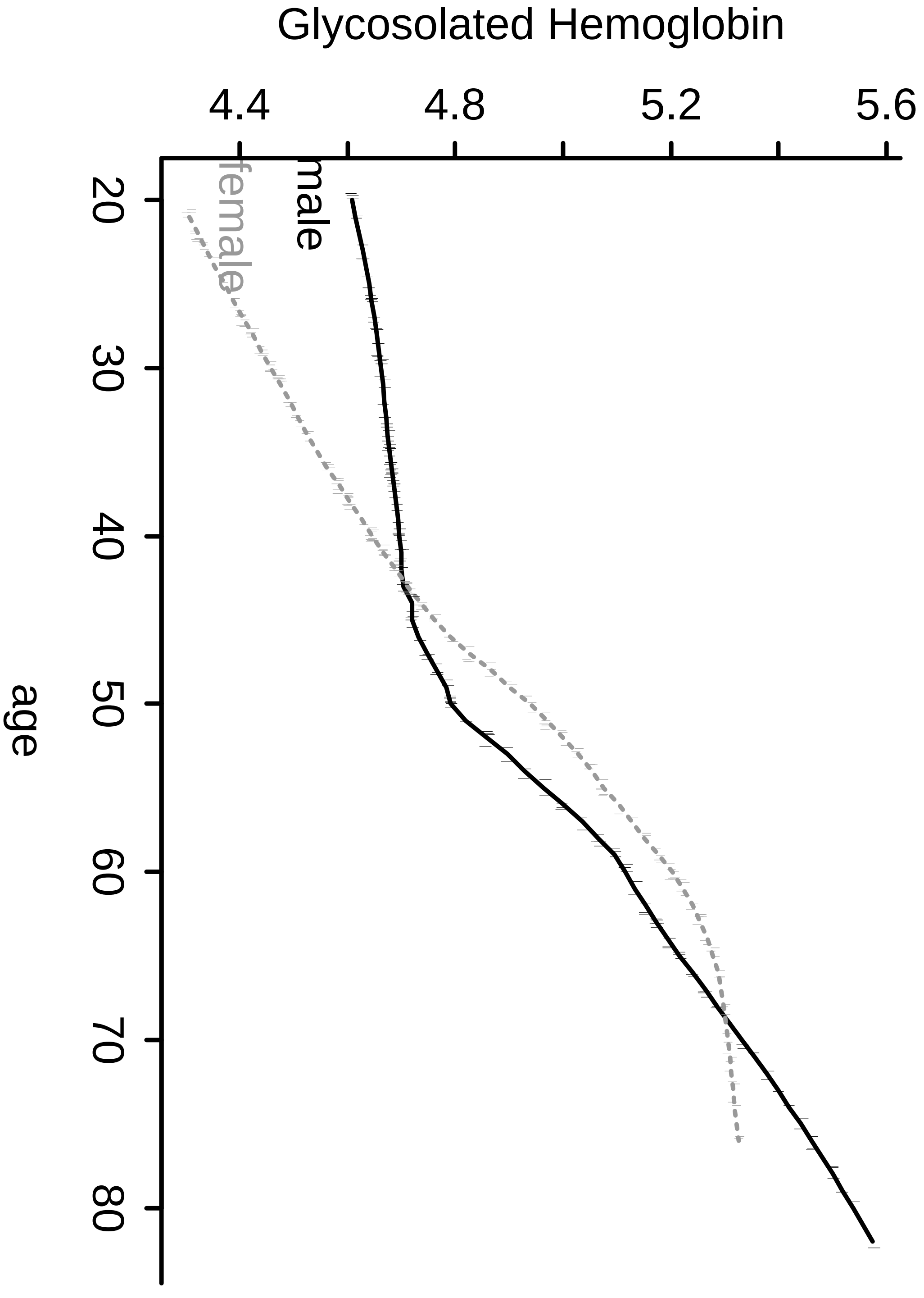


Figure 2.7: *Lowess nonparametric regression of* `glyhb` *vs.* `age`*, stratified by* `gender`

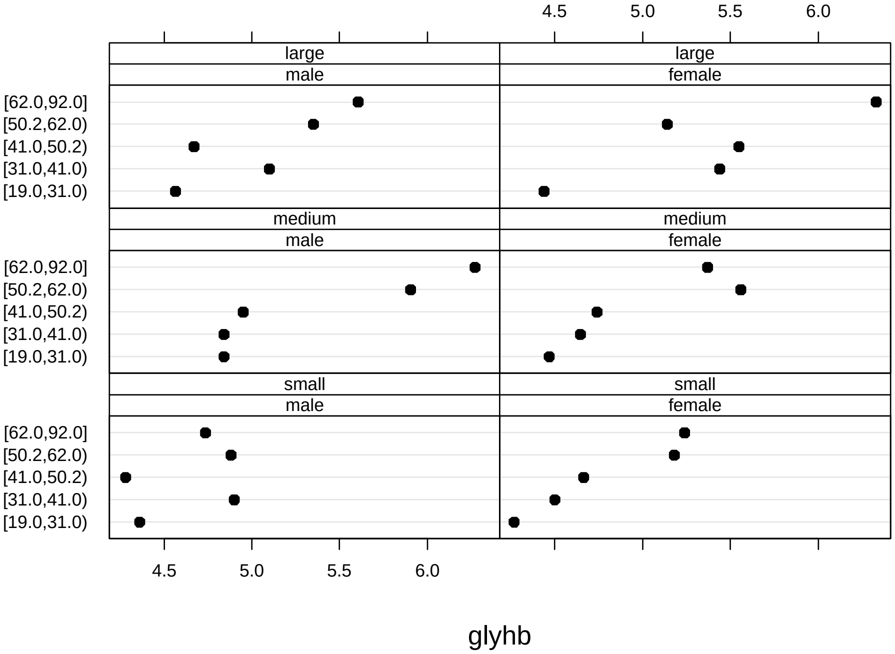


Figure 2.8: *Median* `glyhb` *stratified by* `gender`*,* `frame`*, and quintiles of* `age`

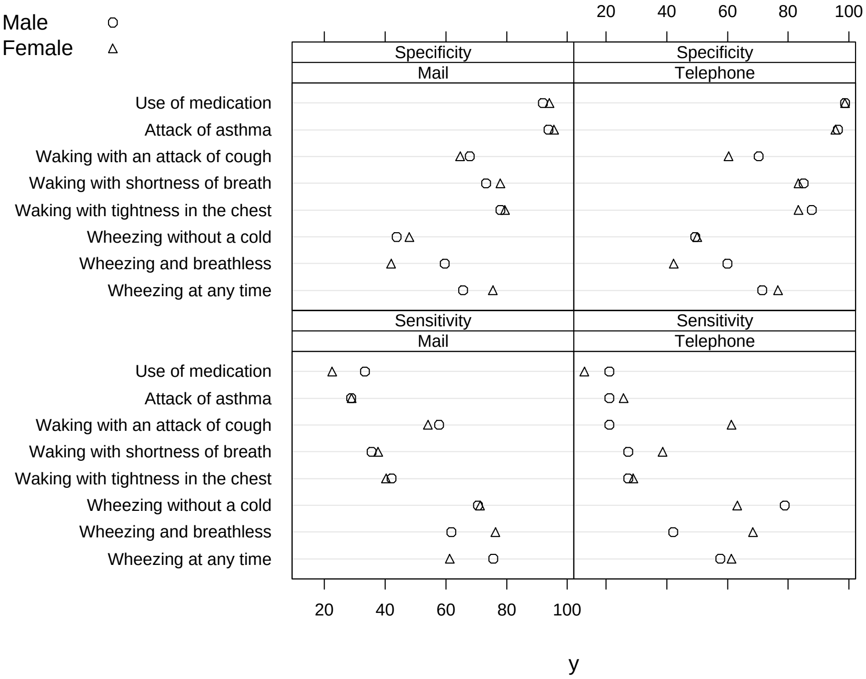


Figure 2.9: *Dot plot depicting sensitivity and specificity stratified by* `method` *and* `sex`*, with the two sexes superposed. Data are from Galobardes,* et al., J Clin Epi *51:875-881, 1998.*

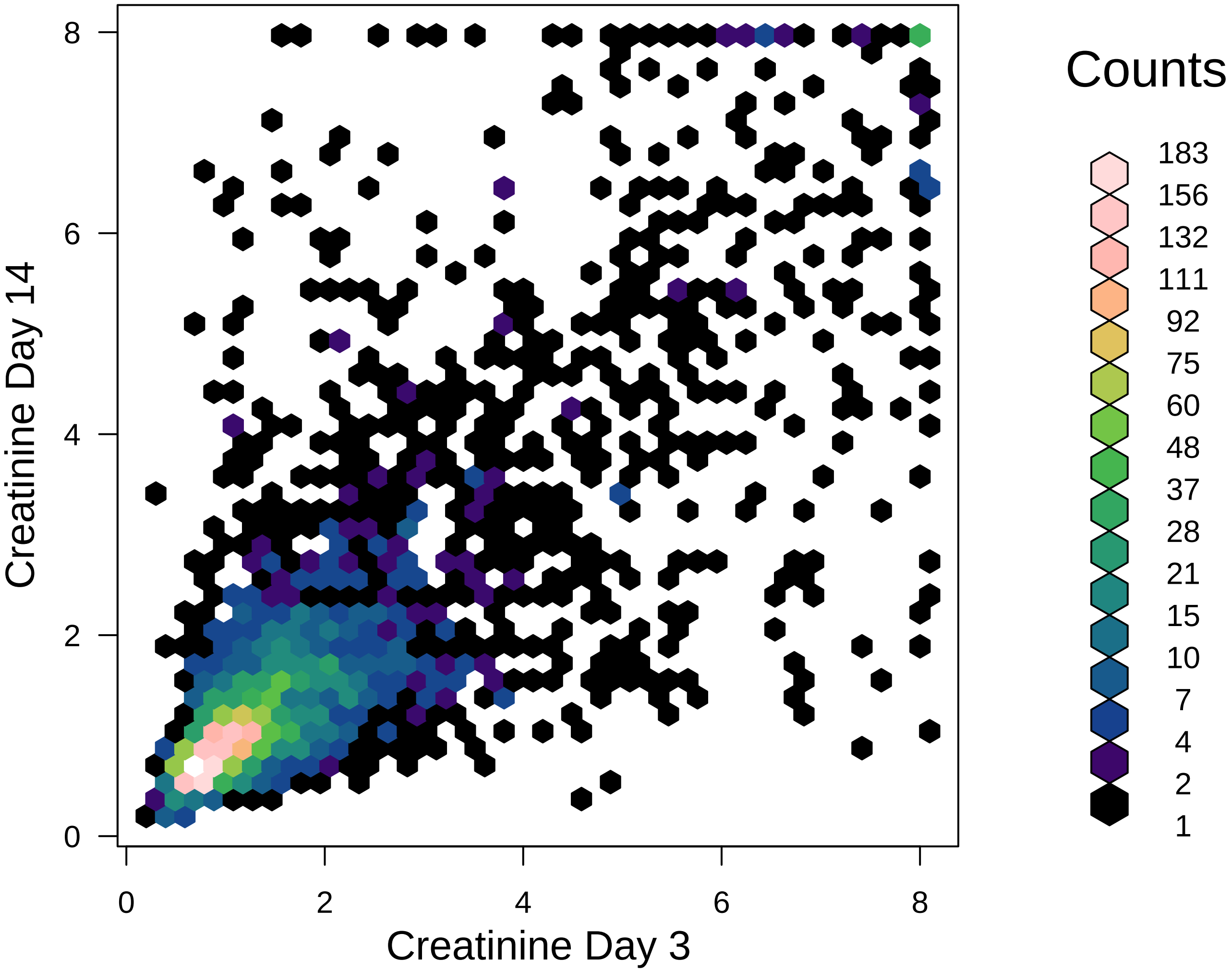


Figure 2.10: *Scatterplot for showing large number of observations using hexagonal binning. Number of observations in each bin is color-coded.*

```
require(hexbin)   # hexbin package on CRAN
h ← hexbin(pmin(8,crea3), pmin(8,crea14), xbins=40)
plot(h, xlab='Creatinine Day 3', ylab='Creatinine Day 14',
     trans=function(x)x^(1/3), inv=function(x)x^3,
     colramp=plinrain)
```

See `http://cran.r-project.org/web/packages/hexbin/vignettes/hexagon_binning.pdf`

2.1.1 Data Provided by Min Wu PhD, Novartis East Hanover

Fasting serum tryglycerides measured longitudinally, multiple treatments. For some displays only use Control and Treatment 1. Patients on Treatment 2 had a lead-in on Treatment 1.

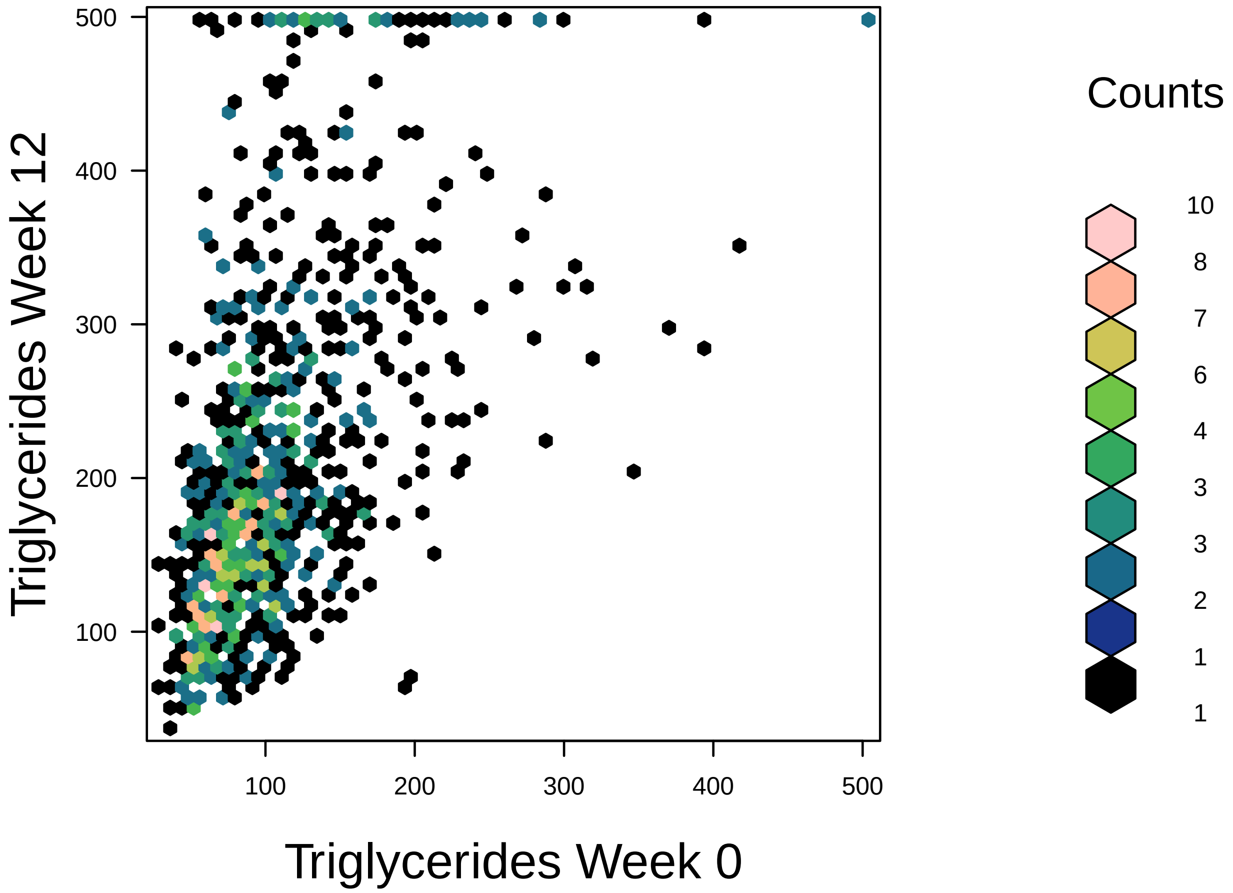


Figure 2.11: *Hexagonal binning plot for week 12 against week 0 triglyceride levels for Treatment 1. Color mapping of sample sizes is on the cube root scale. Measurements are truncated at 500 mg/dl.*

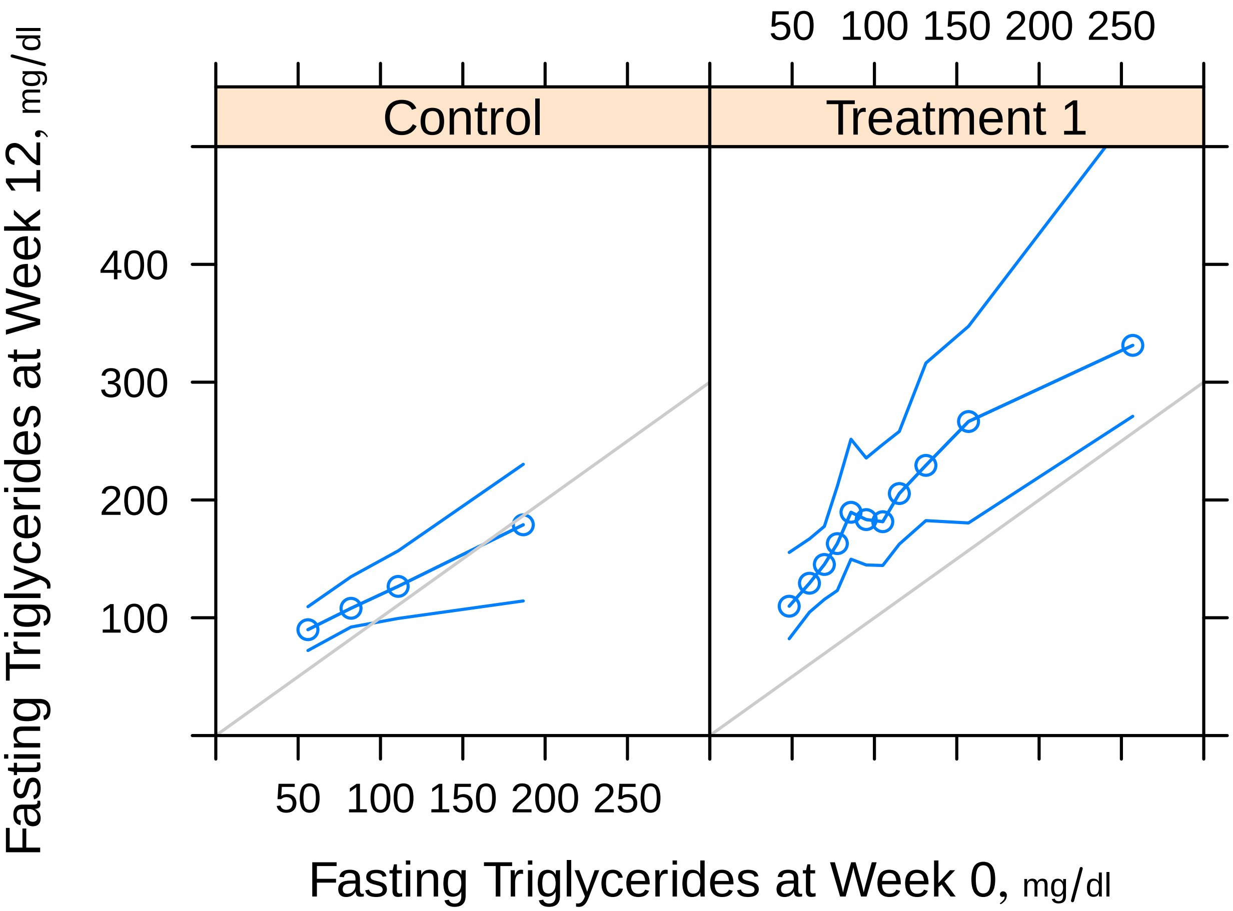


Figure 2.12: *Three quartiles of week 12 triglyceride stratified by baseline triglyceride (100 patients in each interval) and treatment*

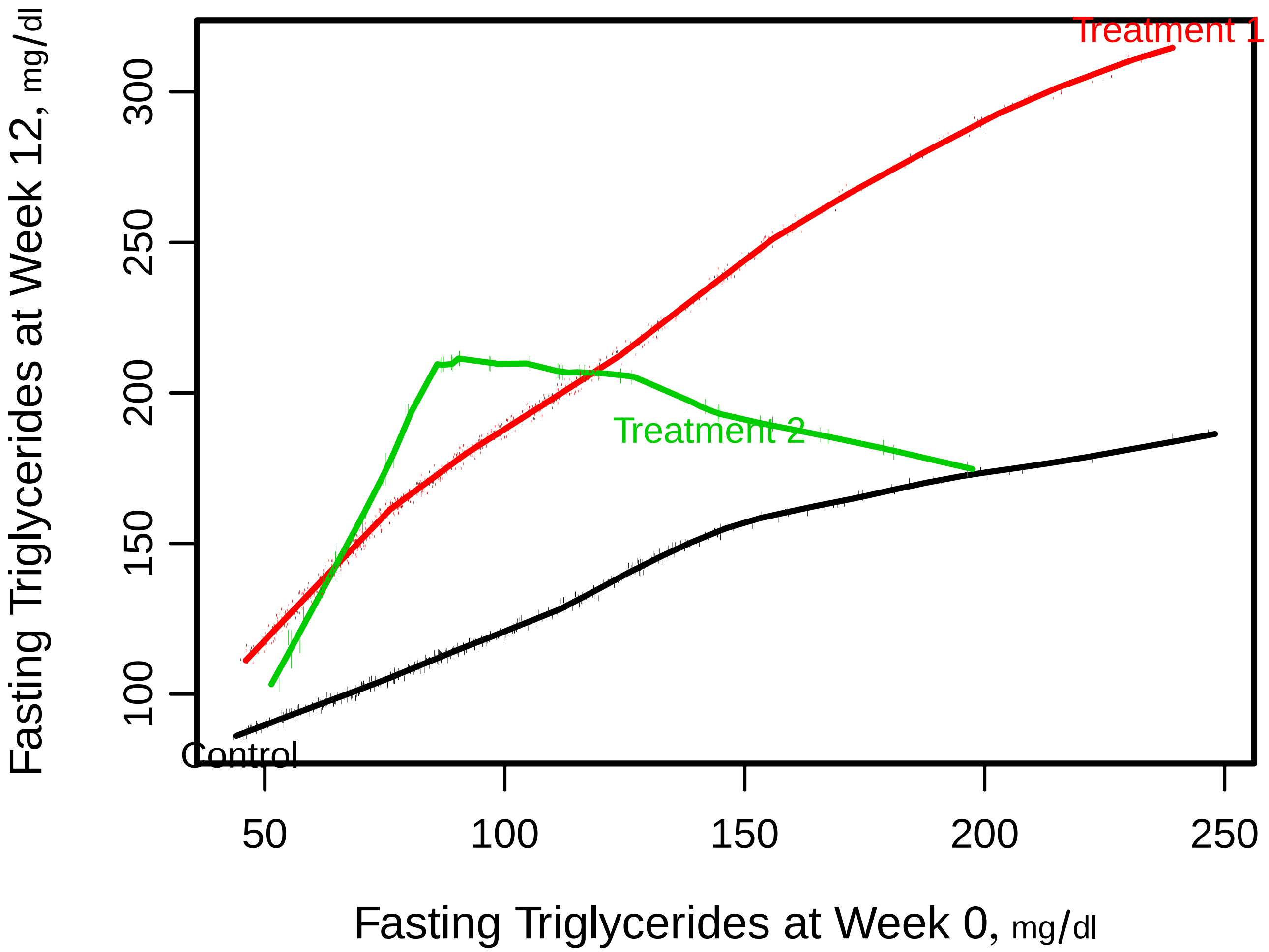


Figure 2.13: loess *smoother of week 12 tryglyceride vs. week 0 stratified by treatment. Rug plots on each curve show distribution of week 0 measurements.*

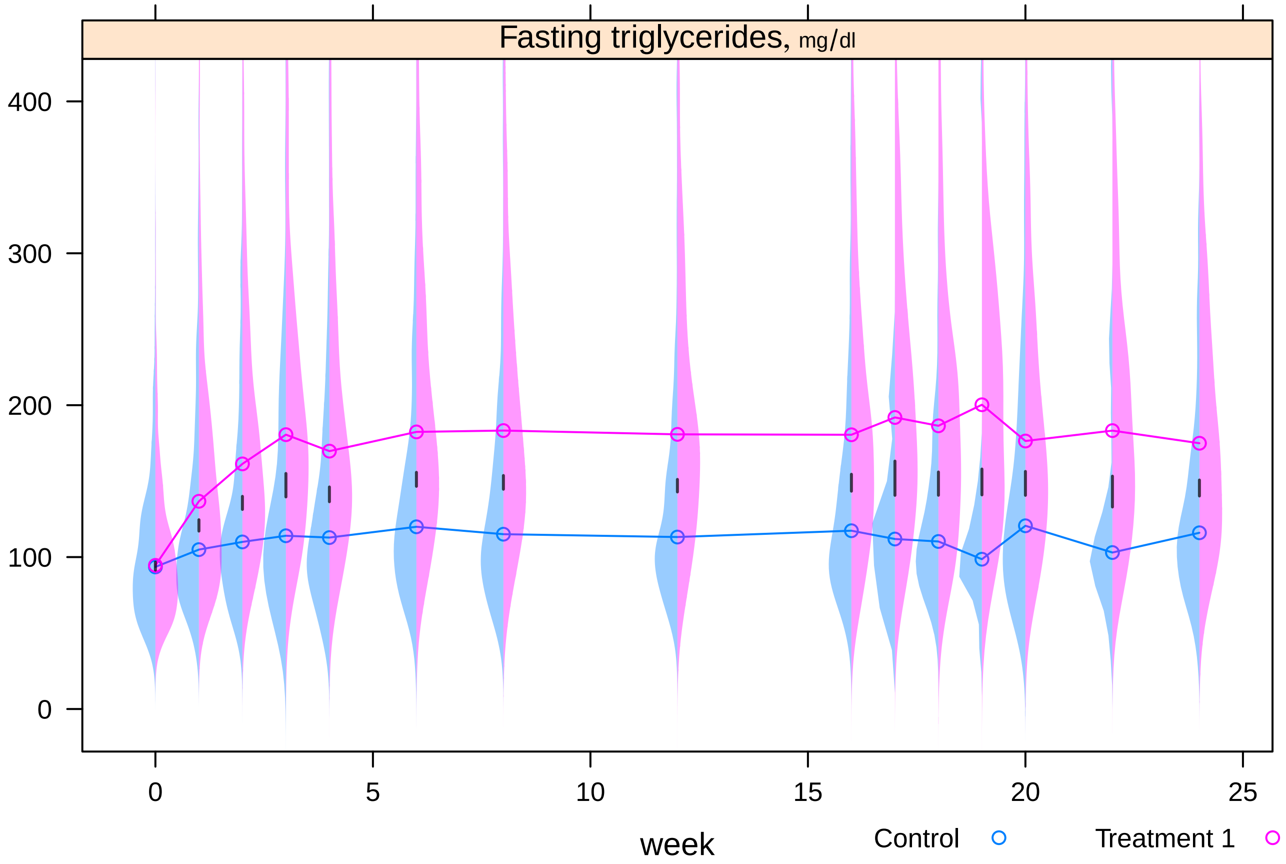


Figure 2.14: *Longitudinal back–to–back violin plots showing the entire distribution of triglyceride by treatment and time. Circles indicate medians. The black vertical bars have lengths equal to one-half the length of the 0.95 confidence interval for the difference in medians. When this bar does not touch the circles, there is a significant difference in medians at the 0.05 level. This type of display is used in the* `greport` *package.*

2.1.2 Displaying Ranks and Confidence Intervals

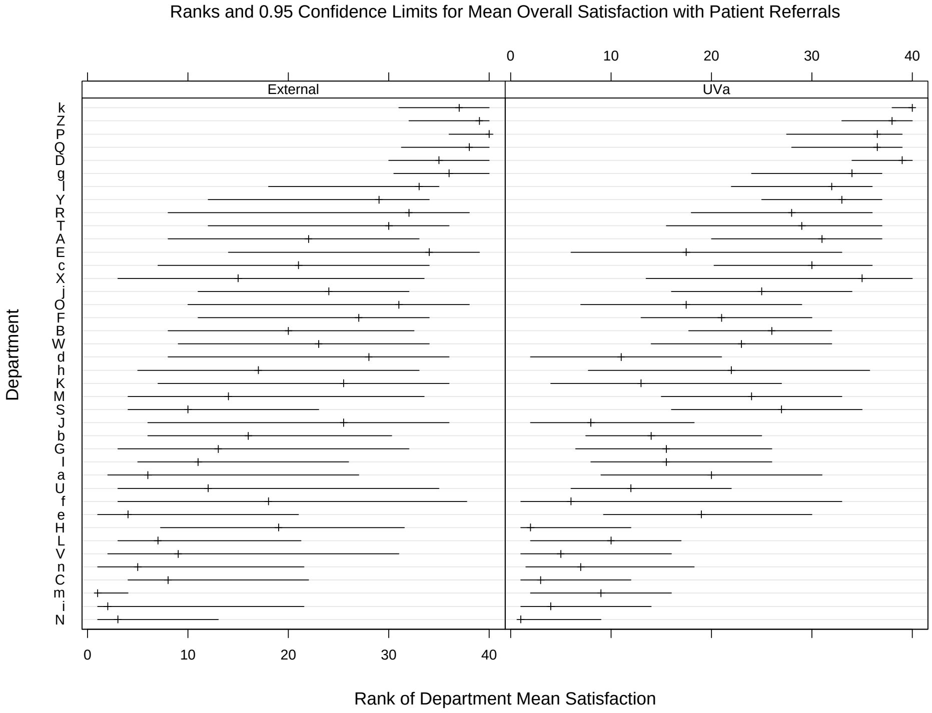


Figure 2.15: *Dot plot showing rank and bootstrap 0.95 confidence limits for the rank of mean satisfaction with service, stratified by UVa vs. outside referring physicians. Dots are sorted by descending order of the **mean** satisfaction across the two strata.*

2.2 Examples from REGRESSION MODELING STRATEGIES, NY: Springer 2001

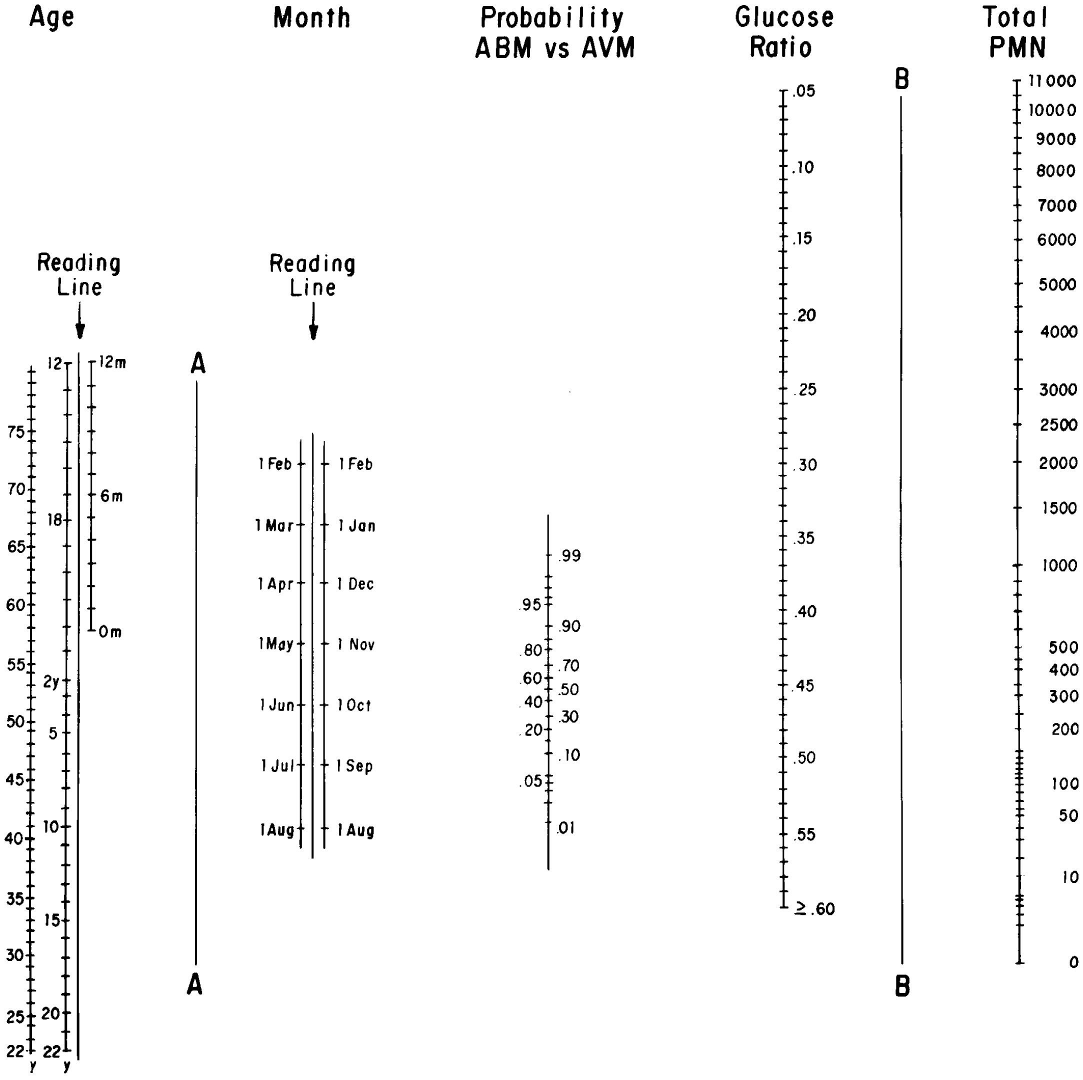


Figure 2.16: *Nomogram for estimating probability of bacterial (ABM) versus viral (AVM) meningitis. Step 1, place ruler on reading lines for patient's age and month of presentation and mark intersection with line A; step 2, place ruler on values for glucose ratio and total polymorphonuclear leukocyte (PMN) count in cerebrospinal fluid and mark intersection with line B; step 3, use ruler to join marks on lines A and B, then read off the probability of ABM versus AVM. Copyright 1989, American Medical Association. Reprinted by permission.*

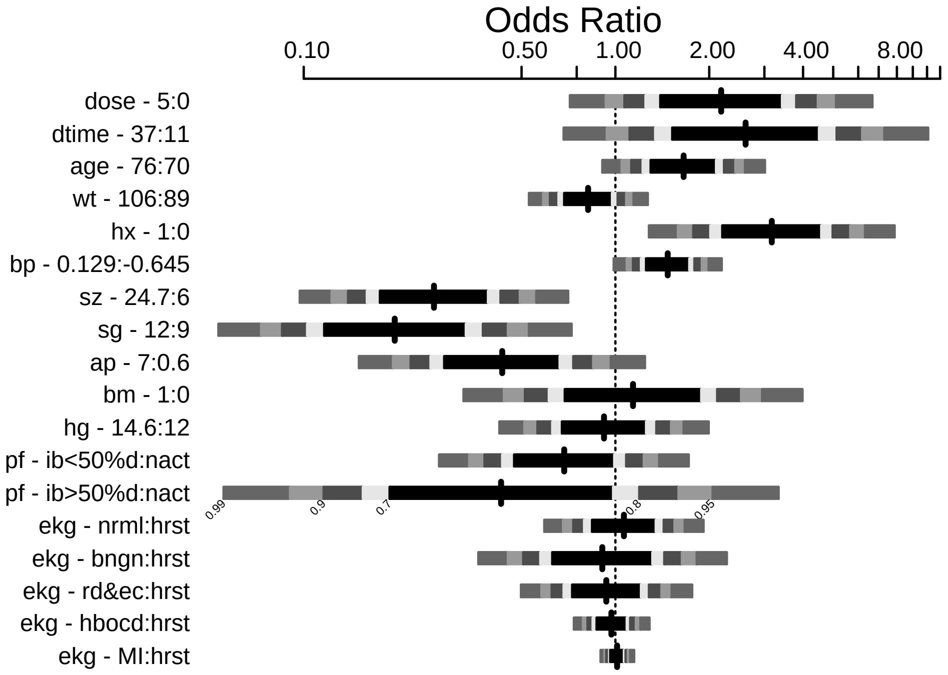


Figure 2.17: *Interquartile-range odds ratios for continuous predictors and simple odds ratios for categorical predictors. Numbers at left are upper quartile : lower quartile or current group : reference group. The shaded bars represent* $0.7, 0.8, 0.9, 0.95, 0.99$ *confidence limits. The intervals are drawn on the log odds ratio scale and labeled on the odds ratio scale. Ranges are on the original scale, even for transformed variables.*

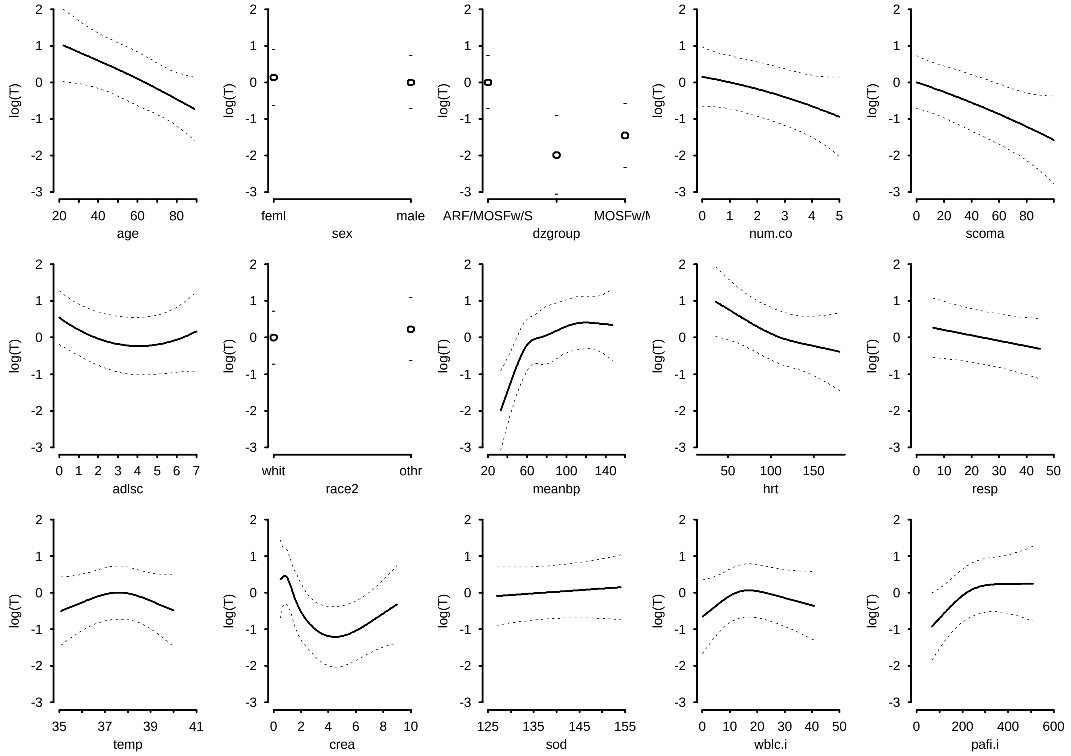


Figure 2.18: *Effect of each predictor on log survival time. Predicted values have been centered so that predictions at predictor reference values are zero. Pointwise* 0.95 *confidence bands are also shown. As all* Y*-axes have the same scale, it is easy to see which predictors are strongest.*

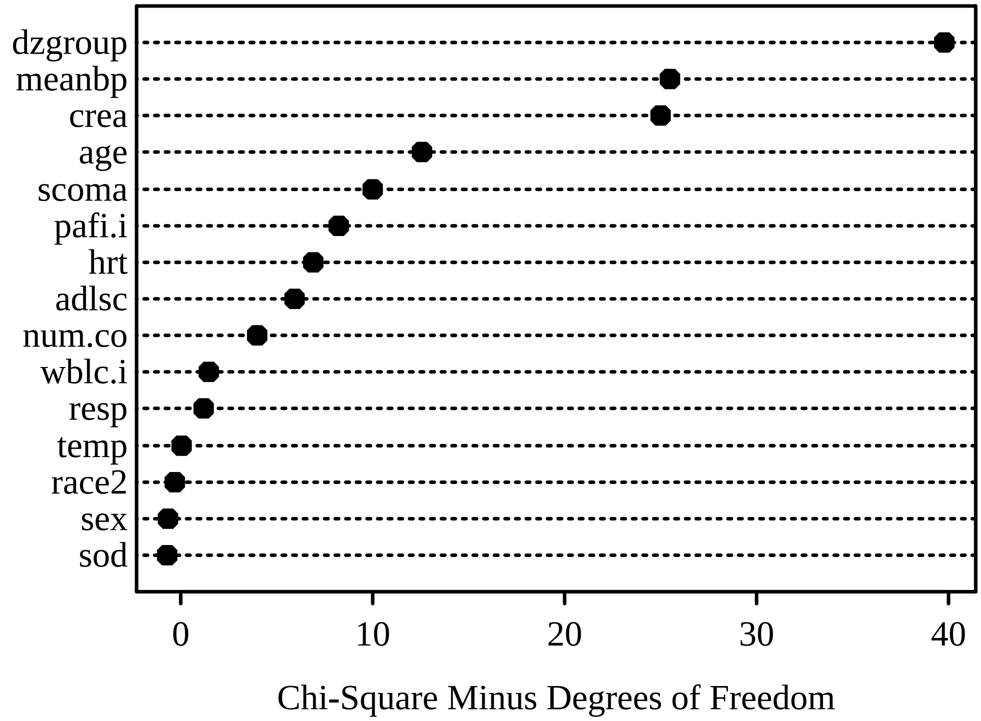


Figure 2.19: *Contribution of variables in predicting survival time in log-normal model.*

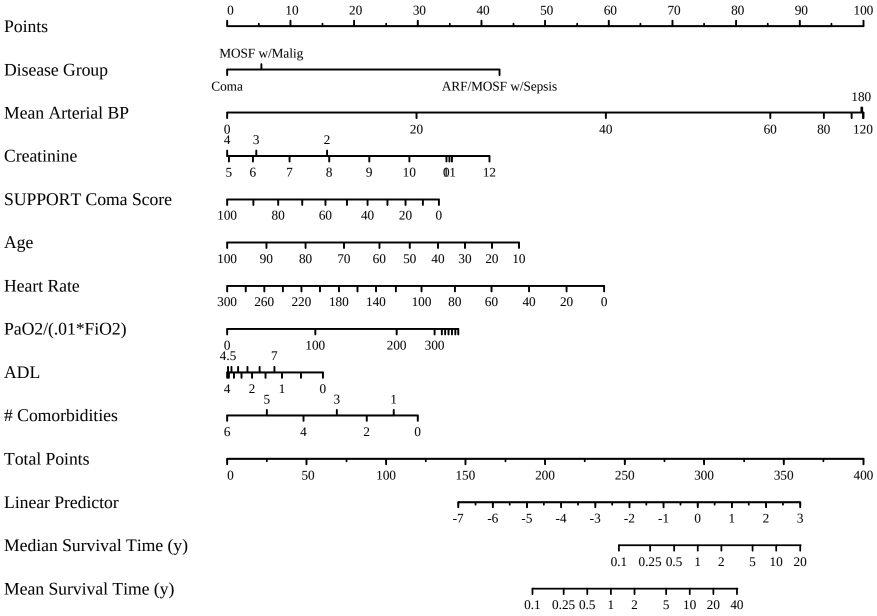


Figure 2.20: *Nomogram for predicting median and mean survival time, based on approximation of full model.*

2.3 Other Devices for Obtaining Predictions

2.3.1 Single-Axis Nomogram

When one needs to transform a variable or to evaluate a prediction model that contains only a single predictor, a single-axis plot may be the most accurate device. In the following example, a binary logistic model of a single continuous predictor variable `Score` is evaluated to obtain the predicted probability of the event of interest.

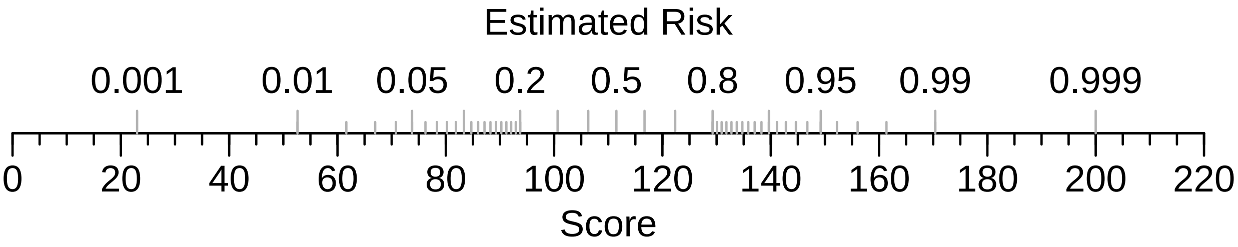


Figure 2.21: *Single-axis nomogram.*

The R code producing this figure is below[a].

```
pdf('oneAxis.pdf', width=6, height=1.25)
par(mar=c(3,.5,1,.5))
beta ← (qlogis(.999)-qlogis(.001))/177
alpha ← qlogis(.001)-23*beta
plot.new()
par(usr=c(-10,230,-.04,1.04))
par(mgp=c(1.5,.5,0))
axis(1, at=seq(0,220,by=20))
axis(1, at=seq(0,220,by=5), tcl=-.25, labels=FALSE)
title(xlab='Score')
```

---

[a] The intercept `alpha` and slope `beta` were computed to match a published model. These would ordinarily be computed from `coef(f)` where `f` is the fit object.

```
p ← c(.001,.01,.05,seq(.1,.9,by=.1),.95,.99,.999)
s ← (qlogis(p)-alpha)/beta
axis(1, at=s, label=as.character(p), col.ticks=gray(.7),
     tcl=.5, mgp=c(-3,-1.7,0))
p ← c(seq(.01,.2,by=.01),seq(.8,99,by=.01))
s ← (qlogis(p)-alpha)/beta
axis(1, at=s, labels=FALSE, col.ticks=gray(.7), tcl=.25)
title(xlab='Estimated Risk', mgp=c(-3,-1.7,0))
dev.off()
```

2.4 Example Pharmaceutical Safety Displays

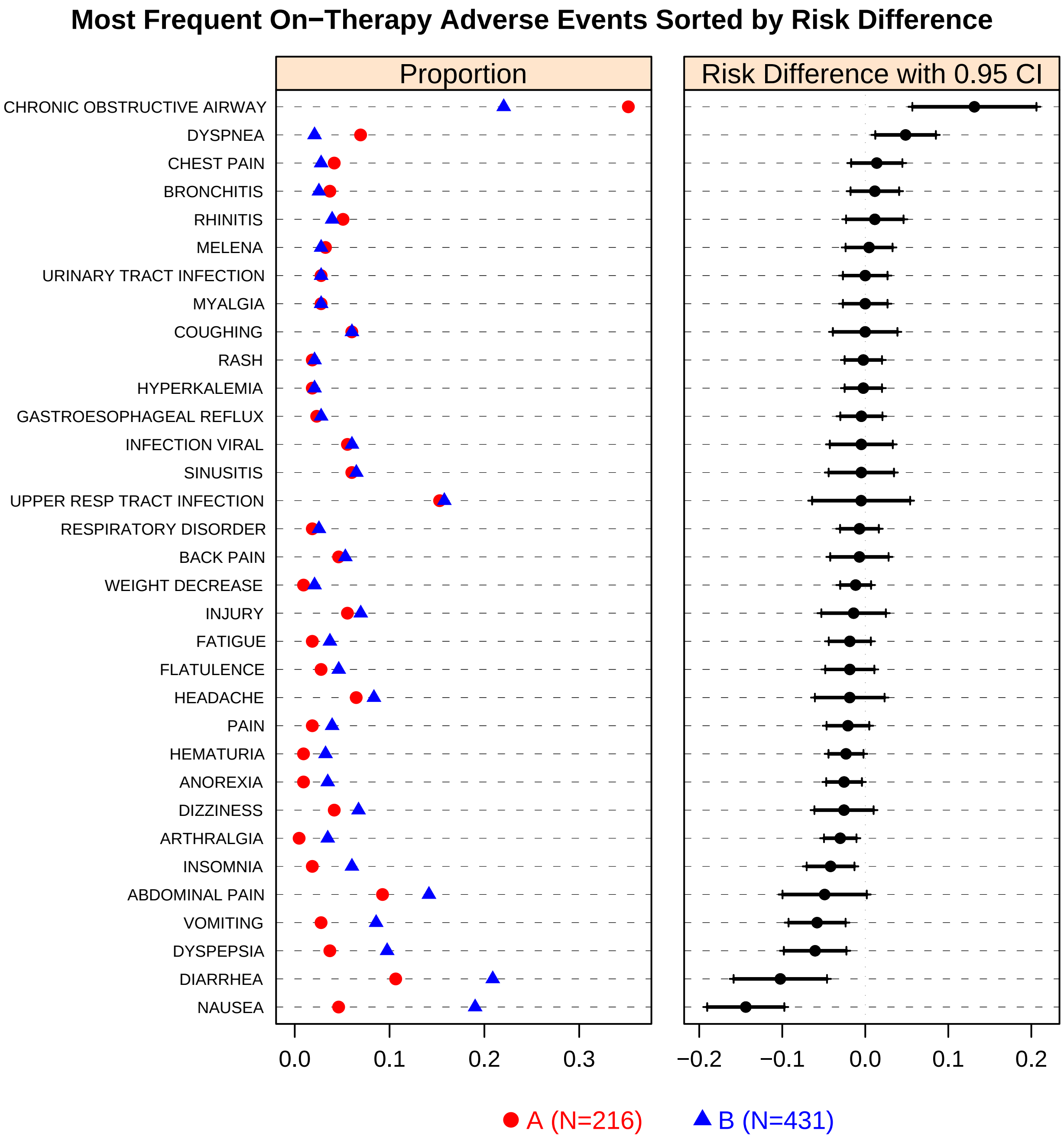


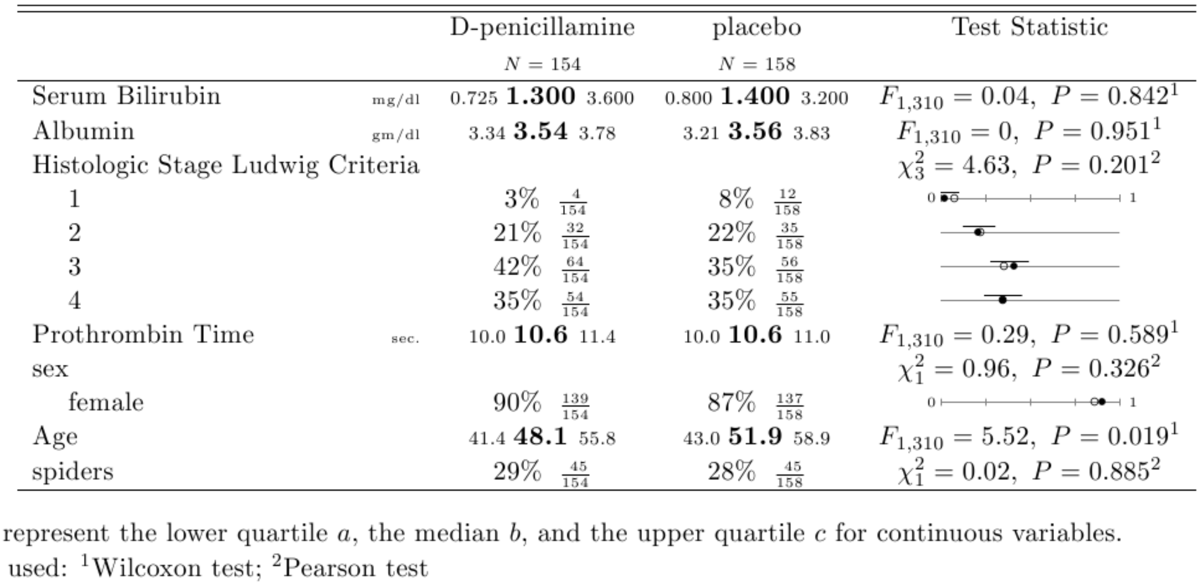
Figure 2.22: *Side-by-side dot plots displaying crude incidence of commonly occuring adverse events in a clinical trial with two treatments (A and B, left panel) and the difference in proportions with 0.95 confidence interval (right panel). AEs (both panels) are sorted by descending difference in proportions. Modification of Amit* et al*, 2008.*

```
## The following functions are in the HH package
panel.ae.dotplot ←
 function (x, y, groups, ..., col.AB, pch.AB, lower, upper)
{
    panel.num ← if.R(s = get("cell", frame = sys.parent()),
        r = panel.number())
    if (panel.num == 1)
        panel.ae.leftplot(x, y, groups = groups,
            col = col.AB, pch = pch.AB, ...)
    if (panel.num == 2)
        panel.ae.rightplot(x, y, ..., lwd = 6, pch = 16,
                           lower = lower, upper = upper)
}

panel.ae.leftplot ← function(x, y, groups, col.AB, ...)
{
    panel.abline(h = y, lty = 2, lwd = 0, col = 1)
    panel.superpose(x, y, groups = groups,
                    col = col.AB, ...)
}

panel.ae.rightplot ←
 function(x, y, ..., lwd = 6, lower, upper)
{
    if.R(r = {
    }, s = {
        panel.segments ← segments
        panel.points ← points
    })
    panel.abline(v = 0, lty = 3, lwd = 0)
    panel.abline(h = y, lty = 2, lwd = 0, col = 1)
    panel.segments(lower, y, upper, y, lwd = 2)
    panel.xyplot(x, y, ..., col = 1, cex = 0.7)
    panel.points(lower, y, pch = 3, col = 1, cex = 0.4)
    panel.points(upper, y, pch = 3, col = 1, cex = 0.4)
```

```
}


## Version of HH's ae.dotplot that uses risk difference and
## uses proportions instead of percents

riskdiff ← function(data, tx='treat', prefterm='PREF',
                    events='SAE', n='SN',
                    conf.int=.95)
  {
    zcrit ← qnorm((1+conf.int)/2)
    i ← order(data[[prefterm]], data[[tx]])
    data ← data[i,]

    N     ← data[[n]]
    prop  ← data[[events]]/N
    pt    ← data[[prefterm]]
    treat ← data[[tx]]
    if(nrow(data) != 2*length(unique(pt)))
     stop('asymmetric data')
    utrt ← sort(unique(as.character(treat)))

    diff ← prop[treat==utrt[1]] - prop[treat==utrt[2]]
    diff.order ← order(diff)
    pt ← ordered(pt, levels=(pt[treat==utrt[1]])[diff.order])
    diff2 ← as.vector(rbind(diff, diff))
    p1 ← prop[treat==utrt[1]]
    n1 ← N[treat==utrt[1]]
    p2 ← prop[treat==utrt[2]]
    n2 ← N[treat==utrt[2]]

    se ← sqrt(p1*(1-p1)/n1 + p2*(1-p2)/n2)
    se2 ← as.vector(rbind(se, se))
    lower ← diff2 - zcrit*se2
    upper ← diff2 + zcrit*se2
    data.frame(pt, treat, prop, diff=diff2, lower, upper)
```

```
}

# To get aeanonym: require(HH);
# aeanonym ← read.table(hh("datasets/aedotplot.dat"), header=T

w ← riskdiff(aeanonym, tx='RAND', prefterm='PREF',
              events='SAE', n='SN')

ae.dotplot ←
  function(data, A.name = 'A', B.name = 'B',
           col.AB = c("red", "blue"), pch.AB = c(16, 17),
           main.title = "Most Frequent On-Therapy Adverse Event
           main.cex = 1, cex.AB.points = NULL,
           cex.AB.y.scale = 0.6,
           position.left = c(0, 0, 0.7, 1),
           position.right = c(0.61, 0, 0.98, 1),
           key.y = -0.2, conf.int=0.95)
{
  r ←
    dotplot(pt ~ prop + diff,
            groups = data$treat,
            data = data, outer = TRUE,
            lower = data$lower,
            upper = data$upper,
            panel = panel.ae.dotplot,
            scales = list(x = list(relation = "free",
                          limits = list(range(data$prop),
                             range(data$lower, data$upper))),
              y = list(cex = cex.AB.y.scale)),
            A.name = A.name,  B.name = B.name,
            col.AB = col.AB, pch.AB = pch.AB,
            cex.AB.points = cex.AB.points,
            cex.AB.y.scale = cex.AB.y.scale,
            main = list(main.title, cex = main.cex),
            xlab = NULL, between = list(x = 1),
            key = list(y = key.y, x = 0.15,
              points = list(col = col.AB, pch = pch.AB),
              text = list(c(A.name, B.name), col = col.AB,
                          cex = 0.9),
              columns = 2, between = 0.5, space = "bottom"))
  r$condlevels[[1]] ← c("Proportion",
                         paste("Risk Difference with",
                          conf.int, "CI"))
  r
}

ae.dotplot(w, A.name='A (N=216)', B.name='B (N=431)')
```

2.5 Mixing Tables and Graphics



		D-penicillamine	placebo	Test Statistic
		$N = 154$	$N = 158$	
Serum Bilirubin	mg/dl	0.725 **1.300** 3.600	0.800 **1.400** 3.200	$F_{1,310} = 0.04,\ P = 0.842^1$
Albumin	gm/dl	3.34 **3.54** 3.78	3.21 **3.56** 3.83	$F_{1,310} = 0,\ P = 0.951^1$
Histologic Stage Ludwig Criteria				$\chi^2_3 = 4.63,\ P = 0.201^2$
1		3% $\frac{4}{154}$	8% $\frac{12}{158}$	
2		21% $\frac{32}{154}$	22% $\frac{35}{158}$	
3		42% $\frac{64}{154}$	35% $\frac{56}{158}$	
4		35% $\frac{54}{154}$	35% $\frac{55}{158}$	
Prothrombin Time	sec.	10.0 **10.6** 11.4	10.0 **10.6** 11.0	$F_{1,310} = 0.29,\ P = 0.589^1$
sex				$\chi^2_1 = 0.96,\ P = 0.326^2$
female		90% $\frac{139}{154}$	87% $\frac{137}{158}$	
Age		41.4 **48.1** 55.8	43.0 **51.9** 58.9	$F_{1,310} = 5.52,\ P = 0.019^1$
spiders		29% $\frac{45}{154}$	28% $\frac{45}{158}$	$\chi^2_1 = 0.02,\ P = 0.885^2$

a b c represent the lower quartile a, the median b, and the upper quartile c for continuous variables.
Tests used: ^{1}Wilcoxon test; ^{2}Pearson test

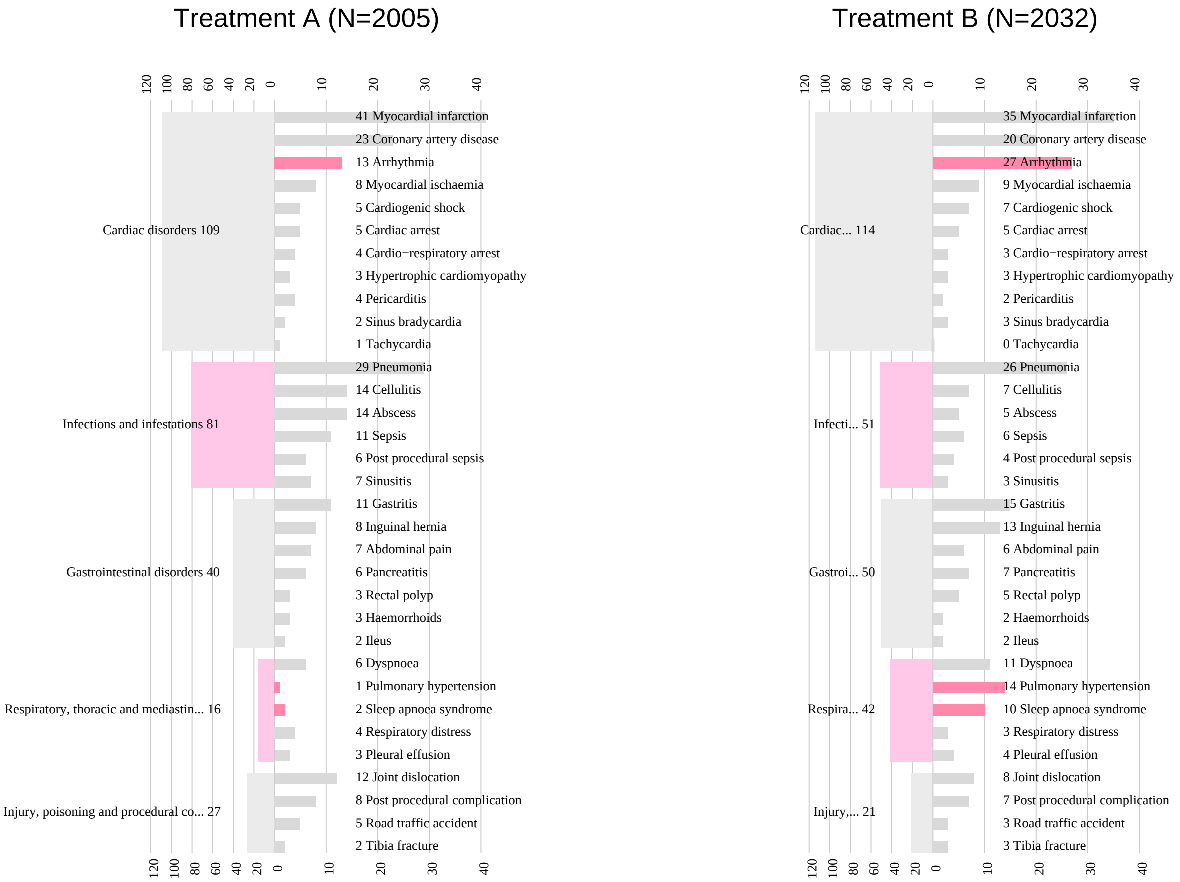


Figure 2.23: *Serious AE frequency display by body system and preferred term. Widths of body system rectangles is proportional to the number of subjects having an event in that body system. The small bars denote the number of subjects who had a particular event. If the between-treatment difference in proportions of subjects having events is significant ($P < 0.05$), the corresponding rectangles/bar charts in both treatment groups are pink/red. Graphic designed and implemented in* R *in the* `rreport` *package by Svetlana Eden, VU Dept. of Biostatistics.*

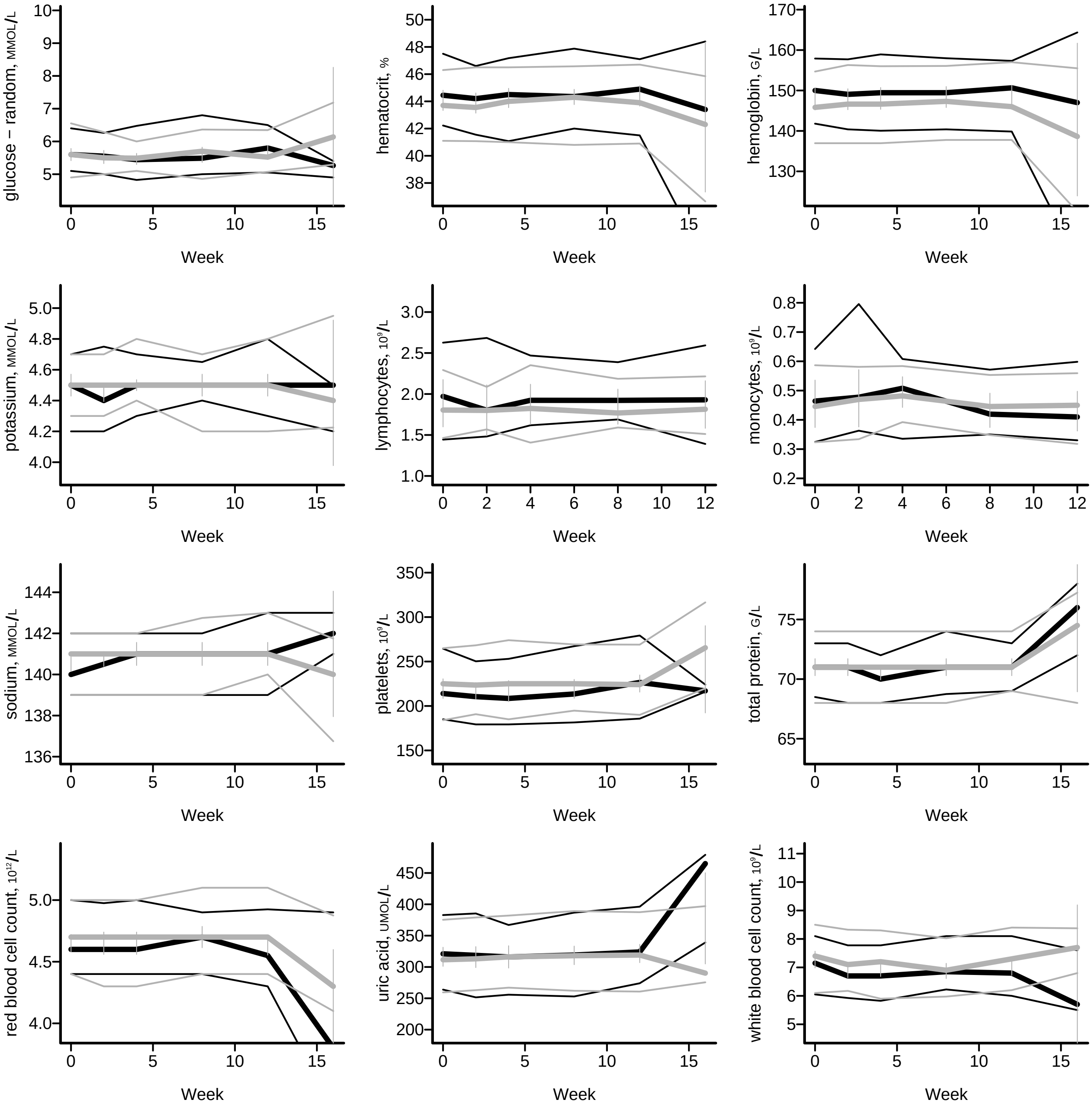
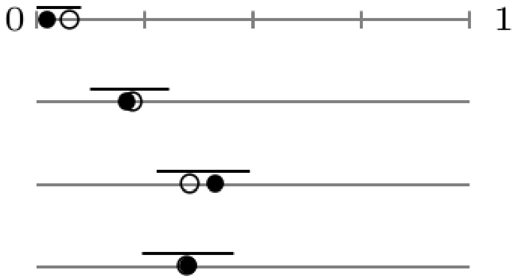


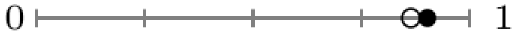
Figure 2.24: *Clinical chemistry quartiles over time for two treatments (one in black, the other in gray scale, thicker lines for medians), with half-width confidence intervals (small vertical lines) for the difference in medians. If the confidence bars do not overlap the two medians, differences are significant at the 0.05 level.*

$$\chi^2_3 = 4.63,\ \ P = 0.201^2$$



$$F_{1,310} = 0.29,\ \ P = 0.589^1$$

$$\chi^2_1 = 0.96,\ \ P = 0.326^2$$



Chapter 3

Graphics for One or Two Variables

Example code in this chapter uses the following functions in the Hmisc package. Other functions are built-in to R.

`datadensity`, `Ecdf`, `histSpike`, `mtitle`, `panel.bpplot`, `pstamp`, `scat1d`, `show.col`, `show.pch`

For more information see

- `http://www.xmarks.com/site/exploringdata.cqu.edu.au`

- `http://davidmlane.com/hyperstat/desc_univ.html`

- `http://www.statsoft.com/Textbook/Graphical-Analyti`

- `http://www.itl.nist.gov/div898/handbook/eda/sectio` `eda15.htm`

- `http://www.math.yorku.ca/SCS/StatResource.html`

3.1 One-Dimensional Scatterplot

- Rug plot; useful by itself or on curves or axes

- Shows all raw data values

- For large datasets, draw random thirds of vertical tick to avoid black blob

- Old-style *dot plots* are similar to rug plots

- Can use Cleveland's dot charts to show raw data

```
rug(x)                        # basic built-in rug plot function
datadensity(mydataframe)      # show 1-d scatterplot for all varia
dotplot(x)                    # one variable
dotplot(∼ x)                  # same thing
```

```
stripplot(x)                 # Trellis/Lattice version
stripplot(∼ x)               # ditto
hist(x)
scat1d(x)                    # add rug plot at top of histogram
plot(x, y)
scat1d(x)                    # rug plot for x at top
scat1d(y, side=4)            # rug plot for y at right side
# scat1d has many options
```

3.2 Histogram

- Used for estimating the *probability density function*

$$f(x) = \lim_{\delta \to 0} \text{Prob}(x - \delta < X \leq x)/\delta \qquad (3.1)$$

- Very dependent on how bins formed, and number of bins
- y-axis can be frequency or proportion
- No statistical estimates can be read directly off a histogram or density plot

```
hist(x, nclass=i)            # use i bins
histogram(x)                 # Trellis/Lattice version
histogram(∼ x)
histSpike(x)                 # high-res spike histogram
plot(x, y)
```

```
histSpike(x, add=T)
# add spike histogram to existing plot, x-axis
```

Note: `histSpike` is called automatically by `scat1d` when n is large (by default, ≥ 2000)

3.3 Density Plot

- Smoothed histogram
- Smooth estimate of $f(x)$ above
- Depends on choice of a smoothing parameter

```
plot(density(x), type='l')
densityplot(~ x)  # Trellis/Lattice version
hist(x, probability=T, nclass=20); lines(density(x)) # ditto
# probability=T scales y-axes so area under curve is 1.0
```

3.4 Empirical Cumulative Distribution Plot

- Population *cumulative distribution function* is

$$F(x) = \mathrm{Prob}(X \leq x) \tag{3.2}$$

- $F(b) - F(a) = \mathrm{Prob}(a < X \leq b)$ and is the area under the density function $f(x)$ from a to b

- Estimate of $F(x)$ is the *empirical cumulative distribution function*, which is the proportion of data values $\leq x$

- Cumulative histogram

- Works fine if histogram has one observation per bin

- ECDF requires no binning and is unique

- Excellent for showing differences in entire distributions between two or three overlaid groups

- Quantiles can be read directly off ECDF

```
Ecdf(mydataframe)                  # show all continuous var
Ecdf(x)
Ecdf(x, q=c(.2,.8))                # ref lines .2,.8 quantiles
Ecdf(x, datadensity='rug')         # add rug plot
Ecdf(x, datadensity='hist')        # add spike histogram
Ecdf(x, datadensity='density')     # add density plot
Ecdf(~ x)                          # Trellis/Lattice version
```

3.5 Box Plot

- Most useful for comparing many groups
- Basically uses 3-number summary: 3 quartiles
- Easy to also show mean
- Can be extended to show other percentiles, especially farther out in the tails of the distribution
- Usually show lower and upper “adjacent” values (“whiskers” and “outside” values; some find these not to be useful

```
boxplot(x)                         # basic function
plot(groups, x)                    # stratified, vertical boxes
boxplot(split(x,groups))           # same
bpplot(split(x,groups))            # box-101 percentile plot
bwplot(x)       # basic horizontal box plot, Trellis/Lattice
bwplot(~ x)                        # ditto
bwplot(x, panel=panel.bpplot)      # horiz. box-percentile plot
```

3.6 Scatter Plots

- Excellent for showing relationship between a semi-continuous X and a continuous Y

- Does not work well for huge n unless relationship is tight
- Can use transformed axes, or transformed data may be plotted
- Can show a limited number of classes of points through the use of different symbols

```
plot(x, y)
plot(x, y, log='xy')                # both axes nonlinear (log)
plot(log(x), log(y))                # log plot, log axes
plot(x, y, main='Main Title')
plot(x, y, xlab='X label', ylab='Y label',
     xlim=c(0,1), ylim=c(20,100))
xyplot(y ∼ x)                       # Trellis / Lattice
```

3.7 Optional Commands to Embellish Non-Trellis Plots

3.7.1 Titles

```
plot(x, y, main='Main Title')
plot(x, y)
title('Main Title')
title(sub='Subtitle', adj=0)
# adj=0,.5,1 for left, center, right-justification
title('First Line\nSecond Line')
# Use \n to jump down one line on output
```

```
par(mfrow=c(2,2),oma=c(0,0,2,0))
# 2x2 matrix of plots, leave 2 lines for
# overall top title (oma=outer margins)
plot( )
hist( )
...
mtitle('Overall Title')
pstamp()                         # date-time stamp lower right
```

3.7.2 Adding Lines, Symbols, Text, and Axes

```
plot(x, y)
axis(3)                          # add axis (ticks & labels)
axis(4, labels=FALSE)            # axis on right (ticks only)
lines(1:3, c(2,4,-1))            # add x=1:3, y=2, 4, -1
points(x2, y2)
points(locator())                # add clicked points
text(.2, 1.3, 'Text')            # add text
text(locator(1), 'Mytext')       # add text at click
```

3.7.3 Reference Lines

```
abline(a=0, b=1)
# line of identity (a,b=intercept,slope)
abline(a=0, b=1, lty=2)          # dotted line
abline(h=c(1,3))                 # horizontal line at y=1,3
abline(v=0)                      # vertical line at x=0
```

3.8 Choosing Symbols, Colors, and Line Types

```
show.pch()                      # display all symbols
points(x, y, pch=i)             # use symbol i from display
show.col()                      # show all colors
points(x, y, col=i)
```

Line types are specified with an `lty` argument. See AH Figure 12.4.

Chapter 4

Conditioning and Plotting Three or More Variables

4.1 Conditioning

- Choose one or two variables of principal interest
  - Typically one for histograms, ECDFs, density plots
  - Two for scatterplots
  - One or two for dot plots
- Can condition on (hold constant) effects of other variables using a statistical model (not covered in this course) or by subsetting data

- Subsets usually non-overlapping for categorical conditioning (stratification) variables

- May or may not be overlapping (shingles) intervals for continuous conditioning variables

- Conditioning may be shown in many ways
  - different symbols or colors for different groups on a scatterplot or dot plot

  - different line styles or colors on a lines plot showing multiple curves, or carefully labeled curves which use the same line styles

  - adjacent lines of dots on a dot plot

  - different vertical, horizontal (or both) panels

  - different pages, including layered transparencies

  - dynamically in real time using “brushing” and other interactive techniques

- Cleveland's principal of small multiples

See Section 1.13 of these lecture notes.

4.2 Dot Plots

- Ideal for showing how one or more categorical variables are related to a single continuous numeric response variable
- Continuous conditioning variables must be categorized
- This is usually done by creating intervals containing equal sample sizes
- Can show error bars and other superpositioning
- Done using Trellis/Lattice `dotplot` or Hmisc Trellis/Lattice `Dotplot` function (later) or using basic `dotchart2` function in Hmisc
- Created automatically when plotting certain objects

created by `summary.formula`

4.3 Thermometer Plots

- Useful in problems that are similar to those handled by dot plots
- But thermometers may be positioned irregularly
- Ideal for geographical displays
- See example from online help file for `symbols`
- For depicting contingency tables see the Hmisc `symbol.freq` function

4.4 Extensions of Scatterplots

4.4.1 Single Plots

- Vary symbols, colors—best for conditioning on categorical variables

- Bubble plots: can depict an addition continuous variable which may be a second *response* variable

- Radius of circles plotted is proportional to the third variable

```
x ← 1:10
y ← runif(10)
z ← runif(10)
symbols(x, y, circles=z, inches=.2)  # largest is .2in
```

4.4.2 Scatterplot Matrices

- Show all pairwise relationships from among 3 or more continuous variables

```
x ← matrix(rnorm(200*5),ncol=5)
pairs(x)
pairs(mydataframe[,c('age','pressure','weight','height')])
```

- Trellis/Lattice function: `splom`

4.5 3-D Plots for Almost Smooth Surfaces

- GUI plus several functions

- Perspective plot: simulated 3-D surface

- Contour plot

- Image plot: 3rd variable categorized into, for example, 10 intervals;
  Shown using color (e.g., heat spectrum) or grayscale
  See main web page for image plot examples

- Basic S functions: `persp, contour, image`

4.6 Dynamic Graphics

4.6.1 Interactively Identifying Points

- If use `plot` or `points`

- Use `identify(x, y, labels=rowlabels)`

```
plot(state.x91[,'Income'], state.x91[,'Murder'])
identify(state.x91[,c('Income','Murder')],
         labels=dimnames(state.x91)[[1]])
```

- First argument to `identify` may be a vector (then 2nd argument is y-variable), a 2-column matrix, or a list

4.6.2 Wireframe and Perspective Plots

- Works for smooth data or somewhat tight relationships

- GUI is nice for this

- Can interactively look at 3-D plot from different perspectives

- Or can automatically get a matrix of plots from varying perspectives

4.6.3 Brushing and Spinning

- Useful for examining relationships between multiple continuous variables when some of the relationships are somewhat tight (depending on the sample size)

- Brushing: highlight points in one 2-D scatterplot; shows corresponding points in other 2-D plots

- Spinning: use motion to simulation 3-D point clouds, rotating 3rd variable in and out of display

```
brush(cbind(sbp, dbp, age, cholesterol), hist=T)
brush(x[,c('sbp','dbp','age','cholesterol')])
brush(x[,Cs(sbp,dbp,age,cholesterol)])
brush(state.x91)      # matrix built-in to S
brush(prim4)          # ''  ''
```

- First argument to `brush` (a matrix) may not contain `NA`s

- `hist` argument: draw marginal histograms

4.6.4 “Live” Graphics on Web Sites

- In R: `shiny`, SVG graphics, javascript-based options

- Allows drilling down to other pre-programmed results

- Simple to use on web sites

R on Web Servers

- Can build web sites at which users click on options, R is run, non-pre-programmed graphics are created on the fly

- R can be freely used on web servers. Information about R may be found at `www.r-project.org`.

- Consider `rstudio.org`

4.7 Lattice Graphics

Function	Purpose	Formula Argument
`barchart`	Bar chart	`y ~ x | g1*g2`
`bwplot`	Box and whisker plot	`y ~ x | g1*g2`
`densityplot`	Probability density plot	`~ x | g1*g2`
`dotplot`	Dot plot	`y ~ x | g1*g2`
`Dotplot`	Hmisc generalization of `dotplot`	`y ~ x | g1*g2`
`Ecdf`	Hmisc ECDF plot	`~ x | g1*g2, groups=g3`
`histogram`	Histogram	`~ x | g1*g2`
`parallel`	Parallel coordinate plot	`~ x | g1*g2`
`panel.bpplot`	Hmisc enhanced box plots and box–%-tile plots with `bwplot`	
`panel.plsmo`	Hmisc `panel` function for `xyplot`	`y ~ x | g1*g2, groups=g3`
`splom`	Multi–panel scatterplot matrices	`~ x | g1*g2`
`stripplot`	One–dimensional scatter plot	`y ~ x | g1*g2`
`xyplot`	Conditioning plots/scatter plots	`y ~ x | g1*g2`
`xYplot`	Hmisc generalization of `xyplot` for multi–column `y`	`Cbind(y,y2,y3) ~ x | g1*g2, groups=g3`
`setTrellis`	Hmisc `trellis` setup	
`trellis.strip.blank`	Hmisc function to set `trellis` to use blank background for panel titles	

General form of first argument (statistical formula):

```
vertical variable ~ horizontal variable | row.conditioner *
   column.conditioner * page.conditioner,
   groups=superposition.variable
```

- Variables after | are conditioned upon to make panels (available for all graph types)

- `groups` variable makes separate lines or symbols within a panel (not available for all graph types)

- All Trellis/Lattice functions take `data` and `subset` arguments.

4.7.1 Appropriate Paneling/Grouping Variables

- These are assumed discrete[a]

- Numeric continuous variables need to be discretized

Panel Variables

- If panel variable is a discrete numeric and you want the value to explicitly show in the panel strip, specify e.g.

```
dotplot(y ~ x | factor(g))
```

- For continuous variable, panels may correspond to overlapping intervals; to create these use `equal.count` or `shingle` functions after the |

---

[a] The GUI will automatically discretize continuous numeric variables when they are used in paneling.

- For non-overlapping intervals, `cut2` is flexible and provides nice panel labeling

4.7.2 Classes of Trellis/Lattice Function

Functions Plotting All Data Points

The following functions are often used on raw data. They are also used to plot summary data computed on raw data, which will be covered later.

- `barchart`
- `dotplot`
- Hmisc version of `dotplot`: `Dotplot`
- `parallel`
- `splom`
- `stripplot`

- `xyplot`

- Hmisc version of `xyplot`: `xYplot`

Functions That Summarize Data and then Plot

- `bwplot`: computes 3 quartiles, mean, outer values, etc. on each group of points (panel, vertical box plot within panel); then draws box plot

- `density`: computes smooth estimate of density function, then plots

- `Ecdf`: computes ECDF for each group or panel (this is an Hmisc function; there is a simpler builtin R function `ecdf`)

- `histogram`: bins x-variable, computes frequencies for each panel

- **Note**: for `density, Ecdf, histogram` user has no control over y-axis as these are computed

- `xYplot` has an argument `method` that can do automatic summarization of raw data; summaries fed immediately into plots

Built-in vs. Hmisc

- For `dotplot`, `xyplot` you must specify `panel=panel.superpose` to use superposition, in addition to specifying `groups`
- `Dotplot`, `xYplot` implicitly handle superposition when `groups` is given
- `Dotplot, xYplot` allow for error bars (`xYplot` also allows for error bands); these are covered later
- `Dotplot, xYplot` use `label` attributes of x and y variables for labeling axes (if not `label` defined, uses variable names as with built-in Trellis/Lattice functions)
- `xYplot` sometimes uses different defaults for the `type` argument
  - User can take control by specifying `type='l','p','b'` for lines, points, or both

- Hmisc functions do some automatic key drawing
- `xYplot` will do some automatic data summarization
- Hmisc has panel functions to be discussed later:
  - `panel.bpplot`: Hmisc function that can be used as a replacement for `panel.bwplot`
  - `panel.plsmo`: can use with `xyplot` to plot lowess trend lines

4.7.3 Panel Functions

- A strength of Trellis/Lattice is its ability to let the user specify a `panel` argument
- This directs Trellis/Lattice in constructing each panel; panel function does not need to know about other panels
- Panel function can be a single function or it can call many panel functions

- Latter is how you combine graph types (e.g., raw data + trend line)

```
xyplot(y ∼ x,
       panel=function(...) {
              panel.xyplot(...)
              panel.loess(...) })
```

- You can more easily get raw data + trend line by using

```
xyplot(y ∼ x, panel=panel.smooth)    # or:
xyplot(y ∼ x, panel=panel.plsmo)     # panel.plsmo in Hmisc
```

Hmisc `panel.bpplot`

- Extends `bwplot` to do box-percentile plots

- By default plots mean using solid circle, and shows 0.25, 0.5, 0.75, and 0.9 coverage intervals, and does not show any raw data

- Has many options

- Examples: `?panel.bpplot`:

```
set.seed(13)
x ← rnorm(1000)
g ← sample(1:6, 1000, replace=T)
```

```
x[g==1][1:20] ← rnorm(20)+3
# contaminate 20 x's for group 1

# default trellis box plot
bwplot(g ∼ x)

# box-percentile plot with data density (rug plot)
bwplot(g ∼ x, panel=panel.bpplot,
       probs=seq(.01,.49,by=.01), datadensity=T)
# add ,scat1d.opts=list(tfrac=1) to make all tick marks
# the same size when a group has > 125 observations

# small dot for means, show only
# .05,.125,.25,.375,.625,.75,.875,.95 quantiles
bwplot(g ∼ x, panel=panel.bpplot, cex=.3)

# suppress means and reference lines for
# lower and upper quartiles
bwplot(g ∼ x, panel=panel.bpplot,
       probs=c(.025,.1,.25), means=F, qref=F)

# continuous plot up until quartiles ("Tootsie Roll plot")
bwplot(g ∼ x, panel=panel.bpplot,
       probs=seq(.01,.25,by=.01))

# start at quartiles then make it continuous
# ("coffin plot")
bwplot(g ∼ x, panel=panel.bpplot,
       probs=seq(.25,.49,by=.01))

# same as previous but add a spike to give 0.95 interval
bwplot(g ∼ x, panel=panel.bpplot,
       probs=c(.025,seq(.25,.49,by=.01)))

# decile plot with reference lines at outer
# quintiles/median
```

```
bwplot(g ~ x, panel=panel.bpplot,
       probs=c(.1,.2,.3,.4), qref=c(.5,.2,.8))

# default plot with tick marks showing all
# observations outside the outer box
# (.05 and .95 quantiles), with very small ticks
bwplot(g ~ x, panel=panel.bpplot,
       nout=.05, scat1d.opts=list(frac=.01))

# show 5 smallest and 5 largest observations
bwplot(g ~ x, panel=panel.bpplot, nout=5)

# Use a scat1d option (preserve=T) to ensure that the
# right peak extends to the same position as the
# extreme
bwplot(~ x , panel=panel.bpplot,
       probs=seq(.00,.5,by=.001),
       datadensity=T, scat1d.opt=list(preserve=T))
```

Hmisc `panel.plsmo`

Lowess nonparametric trend lines (to be discussed later) with enhancements

```
xyplot(y ~ x | year, panel=panel.plsmo, groups=country)
```

Does automatic labeling of curves

4.7.4 Layout and Style Specification

Vertical vs. Horizontal Paneling, Panel Order

- Example: Trellis/Lattice graph with 2 panels
- Default layout is 2 columns, 1 row
- To use 2 rows, 1 column specify

```
trellisfunction(..., layout=c(1,2))
```

- Can also use `layout` just to specify the number of panels:

```
trellisfunction(..., layout=c(3,3))  # 9 panels reserved
```

- Default order is lower left to upper right
- Add `as.table=T` to use LR Top-Bottom ordering

Multiple Trellis/Lattice Plots in One Figure

- Store results of multiple Trellis/Lattice calls in multiple objects

- Use the Trellis/Lattice print method to compose the page

```
p1 ← trellisfunction1(...)
p2 ← trellisfunction2(...)
p3 ← ...
print(p1, split=c(column,row,maxcolumn,maxrow), more=T)
print(p2, split=c(...), more=T)
print(p3, split=c(...), more=F)  # last one
```

4.7.5 Creating Postscript Graphics Files

The Hmisc `setps` function uses decent defaults for B&W graphics

```
setps(plotname, trellis=T, h=...)
trellisfunction( )
dev.off()
```

4.7.6 Controlling Trellis/Lattice Graphical Parameters

- `setps` with `trellis=T` specifies that strip label panels have a blank background for easy reading on black and white graphs

- When making graphs interactively, you can achieve the same effect easily by specifying `trellis.strip.blank()` before creating the graphic. Alternatively, specify an

argument like to following to the Trellis/Lattice function:
`strip=function(...) strip.default(..., style=1)` .
If you have already created a Trellis/Lattice graph you may have to issue `dev.off()` or close the graph sheet window for this to take effect.

- The best way to set up for black and white R Lattice graphics with transparent strips is to put the following commands at the top of the script:

```
library(lattice)
ltheme ← canonical.theme(color = FALSE)
ltheme$strip.background$col ← "transparent"
lattice.options(default.theme = ltheme)
```

- To see a list of arguments that can be specified to the high-level Trellis/Lattice functions type `?trellis.args`. You will see arguments for specifying nonlinear axis scales, panel label strip format, layout, customized keys, axis limits, aspect ratio and banking to 45°, etc.

- To use a $\sqrt{}$ scale, you can specify `scales` as in the following

```
x ← 1:10
y ← x^2
ys ← seq(0,100,by=10)
```

```
xyplot(sqrt(y) ~ x, type='l', ylab='y',
       ylim=sqrt(c(0,100)),
       scales=list(y=list(at=sqrt(ys), labels=format(ys))))
```

- To see many of the current Trellis/Lattice settings, type `show.settings()`

- Type `?trellis.par.get` to learn how to retrieve the current value of any Trellis/Lattice graphical parameter (e.g., line styles, point symbols, dot symbols, strip background, etc.)

- To change a parameter, use `trellis.par.set`

```
dev.off()
# Trellis/Lattice needs to have the device inactive to
# do this
tpl ← trellis.par.get('plot.line')
tpl$lwd ← 3        # change line width to 3
trellis.par.set('plot.line', tpl)
```

This will affect all subsequent Trellis/Lattice graphs. These three commands will for example cause the line thickness of error bars drawn by the `Dotplot` function to be 3 instead of the default of 1.

4.7.7 Summarizing Data for Input to Trellis/Lattice Functions

- Most frequently, summarizations for Trellis/Lattice use simultaneous cross-classification, unlike `summary(..., method=’response’)`

- Hmisc `summarize` function is made for this

- Produces a ready-to-use data frame that will appear to a Trellis/Lattice function to be raw data

```
set.seed(111)
dfr ← expand.grid(month=1:12, year=c(1997,1998),
                  reps=1:100)
attach(dfr)
y ← abs(month-6.5) + 2*runif(length(month)) + year - 1997
s ← summarize(y, llist(month,year), mean, na.rm=T)
s
xYplot(y ∼ month, groups=year, type='b', data=s)
```

- To compute proportions, take means of binary variables

```
s ← summarize(y > 6, llist(month,year), mean,
              stat.name='ygt6', na.rm=T)
s
xYplot(ygt6 ∼ month | factor(year), type='b', data=s)
```

- `FUN` (3^{rd}) argument to `summarize` may specify a function that computes multiple statistics

- This is used to make error bars and bands

4.7.8 Error Bars and Bands

- Used for
  - Measures of precision: $\pm$ S.E., $\pm 2\times$ S.E., (possibly asymmetric confidence limits for a population mean)
  - Measures of variability of raw data: $\pm 2\times$ S.D., quantiles
- Think of upper and lower values as 2^{nd} and 3^{rd} response variables
- Trellis/Lattice allows only a univariate response variable
- Hmisc tricks Trellis/Lattice by using the Hmisc `Cbind` function to "hide" the upper and lower values (and possibly more) in an attribute `'other'` to a single response variable
- Hmisc `xYplot` and `Dotplot` functions allow for such multiple response variables

- Functions such as `smean.cl.normal`, `smedian.hilow`, `smean.sdl` (Hmisc) are set up to create the central value (e.g., mean) and variables named `Lower` and `Upper`

- `FUN` argument of `summarize` can use these functions nicely with `xYplot` and `Dotplot`

`xYplot`

- If you have already computed the lower and upper values (or the S.E.) you can give these directly to `xYplot`:

```
xYplot(Cbind(y,lower,upper) ~ month)
xYplot(Cbind(y,2*se) ~ month)
```

  In the latter example, `y-2*se` and `y+2*se` are automatically computed (because there are only 2 arguments to `Cbind`).

- Note: In standard R Lattice package you can add error bars to plots with `xyplot` by passing an auxiliary variable and using the `subscripts` of the data being plotted in the current panel[b]:

```
 xyplot(y ~ x, data, sd = data$sd,
        panel = function(x, y, subscripts, sd, ...) {
            larrows(x, y - 2 * sd[subscripts],
```

---

[b] Example provided by Deepayan Sarkar on `r-help`, 6Sep04.

```
                   x, y + 2 * sd[subscripts],
                   angle = 90, code = 3, ...)
           panel.xyplot(x, y, ...)
       })
```

- More often we compute summaries to plot, e.g.:

```
# The following example uses the summarize function
# in Hmisc to compute the median and outer quartiles.
# The outer quartiles are displayed using "error bars"
set.seed(111)
dfr ← expand.grid(month=1:12, year=c(1997,1998),
                   reps=1:100)
attach(dfr)
y ← abs(month-6.5) + 2*runif(length(month)) + year-1997
s ← summarize(y, llist(month,year),
               smedian.hilow, conf.int=.5)
xYplot(Cbind(y,Lower,Upper) ∼ month, groups=year, data=s,
       keys='lines', method='alt')    # Figure 4.1

# Can also do:
s ← summarize(y, llist(month,year), quantile,
               probs=c(.5,.25,.75),
               stat.name=c('y','Q1','Q3'))
xYplot(Cbind(y, Q1, Q3) ∼ month, groups=year,
       data=s, keys='lines')
```

- To display means and bootstrapped nonparametric confidence intervals:

```
s ← summarize(y, llist(month,year), smean.cl.boot)
s
 month year    y Lower Upper
     1 1997 6.55  6.44  6.67
```

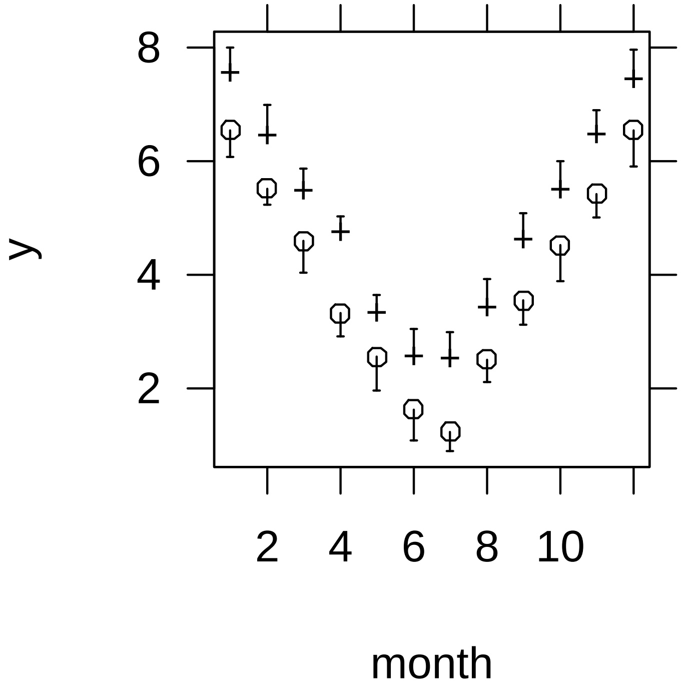


Figure 4.1: *Alternating error bars showing quartiles of raw data.*

```
     1 1998 7.51  7.40  7.62
     2 1997 5.58  5.47  5.69
     2 1998 6.44  6.33  6.55
     . . . . . .
    12 1998 7.47  7.37  7.58

xYplot(Cbind(y, Lower, Upper) ∼ month | factor(year),
       type='l', data=s)
# Figure 4.2
# factor(year) causes year to be written in panel labels

# Can also use Y ← cbind(y, Lower, Upper)
#              xYplot(Cbind(Y) ∼ ...)
# Or:
xYplot(y ∼ month | year, nx=F, method=smean.cl.boot)
# see later
```

```
xYplot(Cbind(y, Lower, Upper) ∼ month | factor(year),
       method='filled bands', type='l', data=s)
# Figure 4.3
# Use method='bands' for ordinary unfilled bands
```

- Here is an example using double bands, to depict the following quantiles: $.1$ $.25$ $.5$ $.75$ $.9$. The 0.25 and 0.75 quantiles are drawn with line thickness 2, and the central line with a thickness of 4. Note that summarize produces a matrix for `y` when `type='matrix'` is specified, and `Cbind(y)` trusts the first column to be the point estimate (here the median)

```
 s ← summarize(y, llist(month,year), quantile,
               probs=c(.5,.1,.25,.75,.9),
               type='matrix')
```

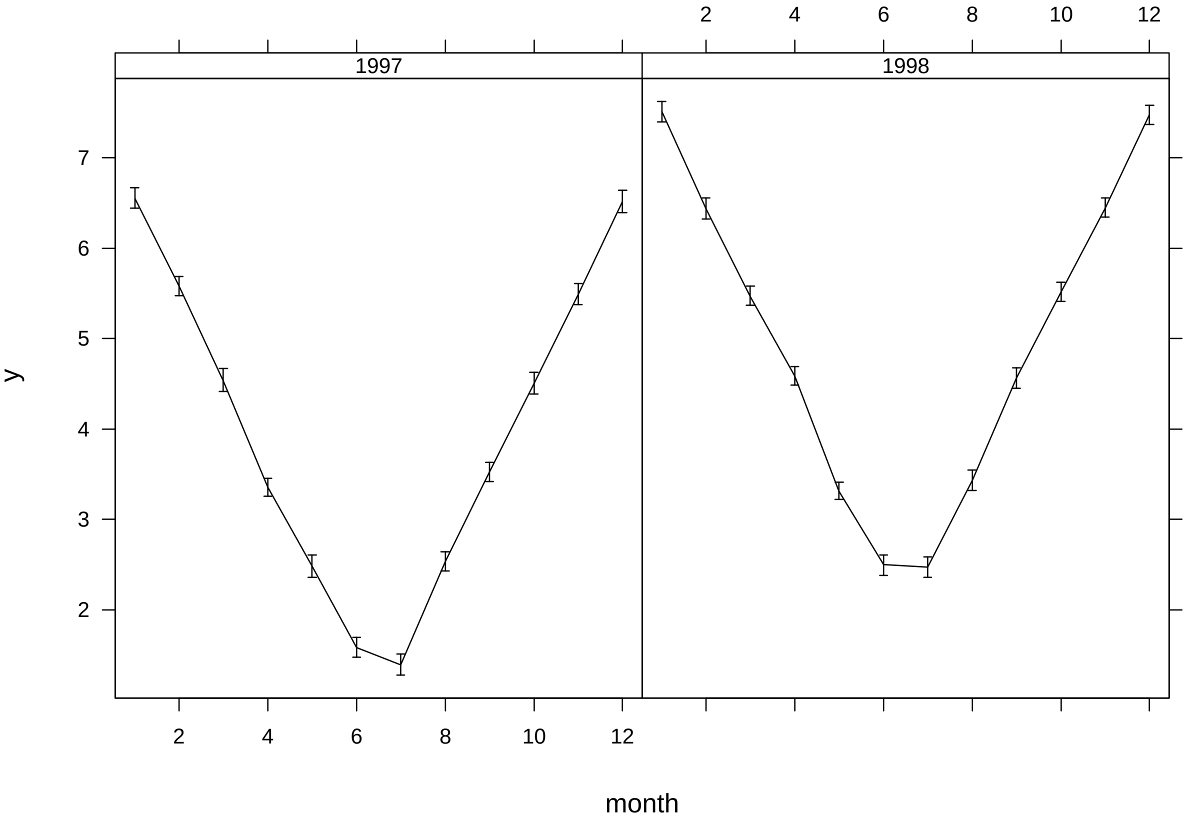


Figure 4.2: *Mean and nonparametric bootstrap 0.95 confidence intervals*

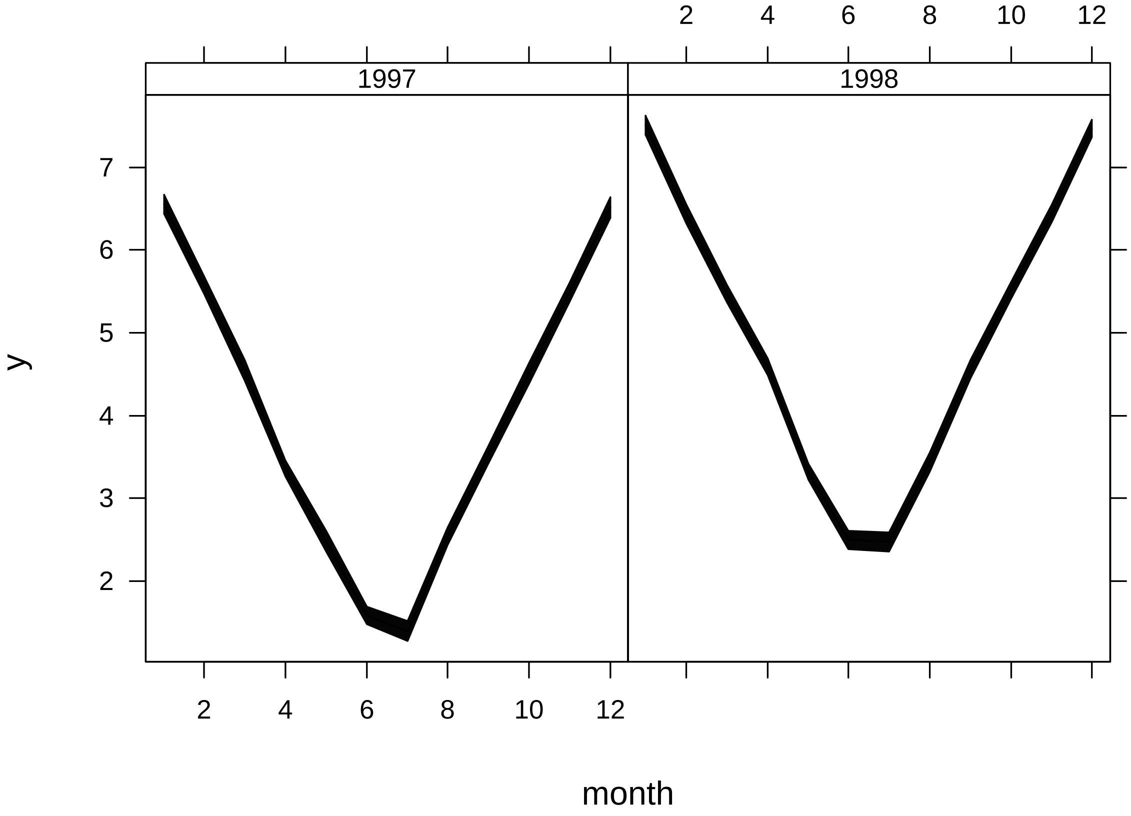


Figure 4.3: *Nonparametric bootstrap confidence limits for each month, but depicted with filled bands*

```
 xYplot(Cbind(y) ~ month | factor(year), data=s,
        type='l', method='bands', lwd.bands=c(1,2,2,1),
        lwd=4)
# Figure 4.4
```

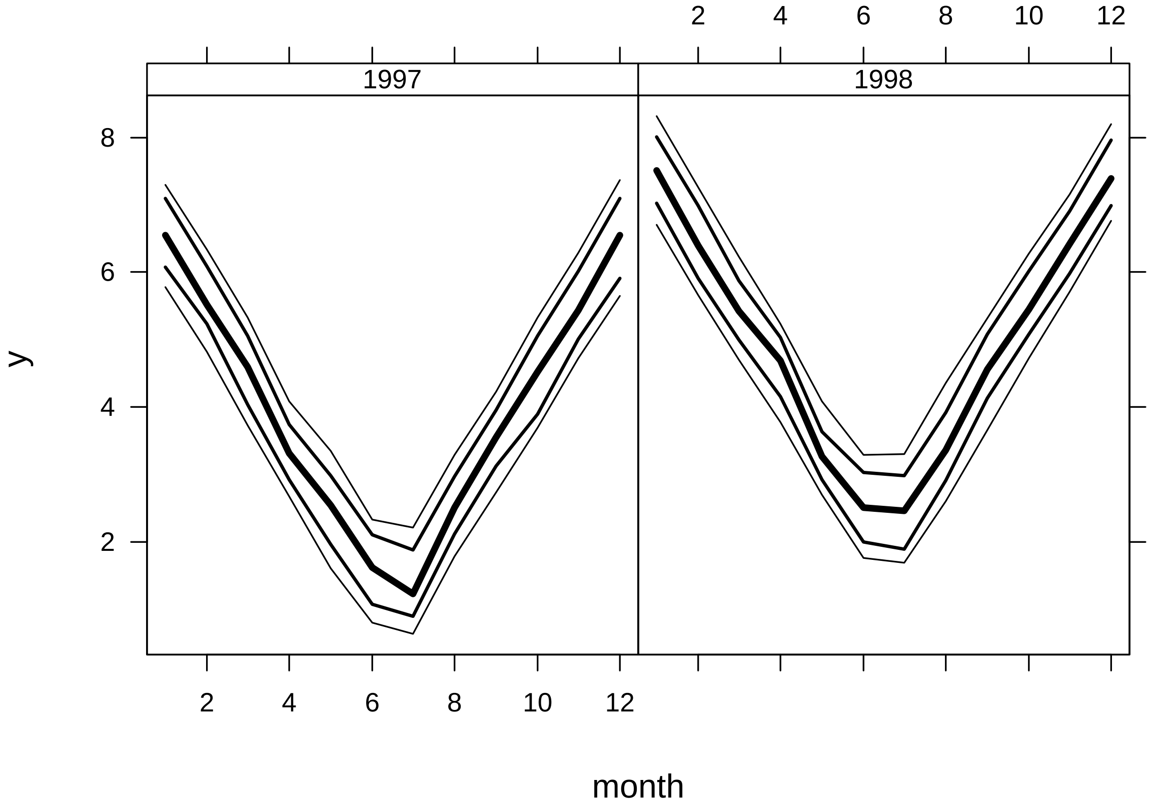


Figure 4.4: *Central line depicts the median, and bands depict the* $0.1, 0.25, 0.75, 0.9$ *quantiles of the raw data*

`xYplot` **with** `method='quantile'` **or** `method=functionname`

- `method='quantile'`: `xYplot` automatically groups the `x` variable into intervals containing a target of `nx` observations

- Default value of `nx` is the lesser of 40 and $\frac{1}{4}\times$ stratum size (specify `nx=0` to do no grouping; useful when `x` variable is discrete such as `month`)

- Quantiles given by the `probs` argument; default is `probs=c(.5,.25,.75)`

- Within each `x` group computes three quantiles of `y` and plots these as three lines

- Mean `x` within each `x` group is taken as the `x`-coordinate

- Useful empirical display for large datasets in which scatterdiagrams are too busy to see patterns of central tendency and variability; good for residual plots for showing symmetry and lack of trend in central tendency and variability[c]

- Can also specify a general function of a data vector that returns a matrix of statistics for `method`; the statistic in the first column should be the measure of central tendency

---

[c] Specify `method=smean.sdl` to instead plot mean and $\pm 2\times$ S.D.

- Arguments can be passed to that function a list `methodArgs`

- Example: Group `x` into intervals containing 40 observations, plot the $0.5, 0.25, 0.75$ quantiles of `y` against mean `x` in interval

```
set.seed(1)
age ← rnorm(1000, 30, 10)
sbp ← 0.3*(age-30) + rnorm(1000, 120, 15)
xYplot(sbp ~ age, method='quantile',     # Figure 4.5
       xlim=c(5,60), ylim=c(100,140))
```

- Instead of quantiles of raw data, show parametric confidence bands, and require 60 observations in a group

```
xYplot(sbp ~ age, method=smean.cl.normal, # Figure 4.6
       xlim=c(5,60), ylim=c(100,140), nx=60)
```

`Dotplot`

- “Multivariate response” packaged by `Cbind` appears as the `x`-variable after the $\sim$

- Does not work well with superposition of `groups`

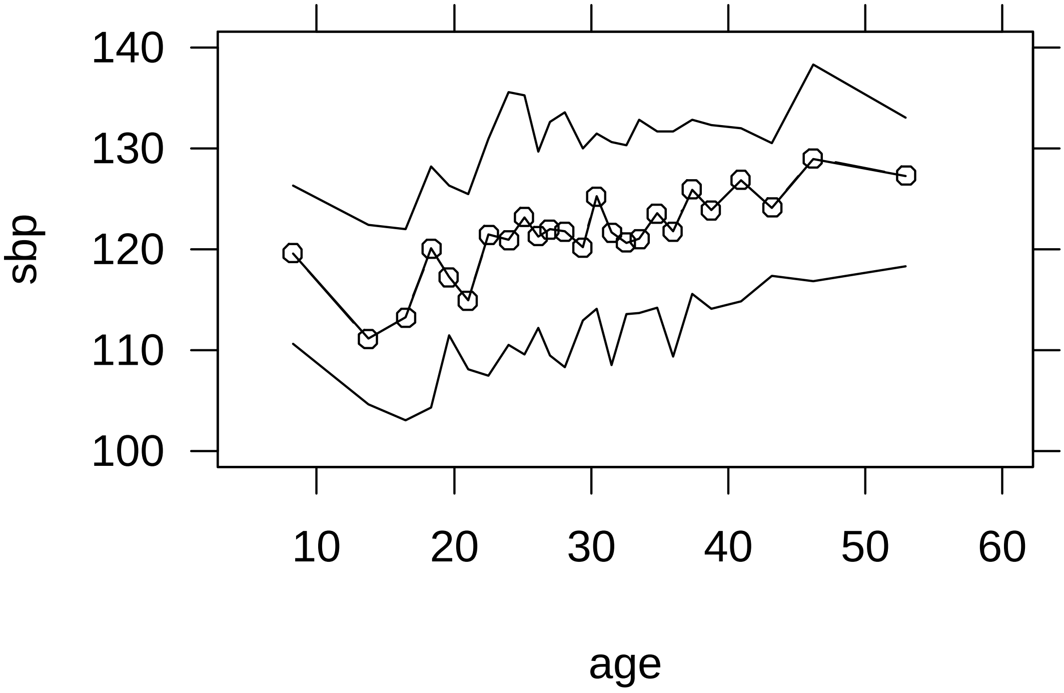


Figure 4.5: $0.25, 0.5, 0.75$ *quantiles of* `sbp` *vs. intervals of* `age` *containing 40 observations*

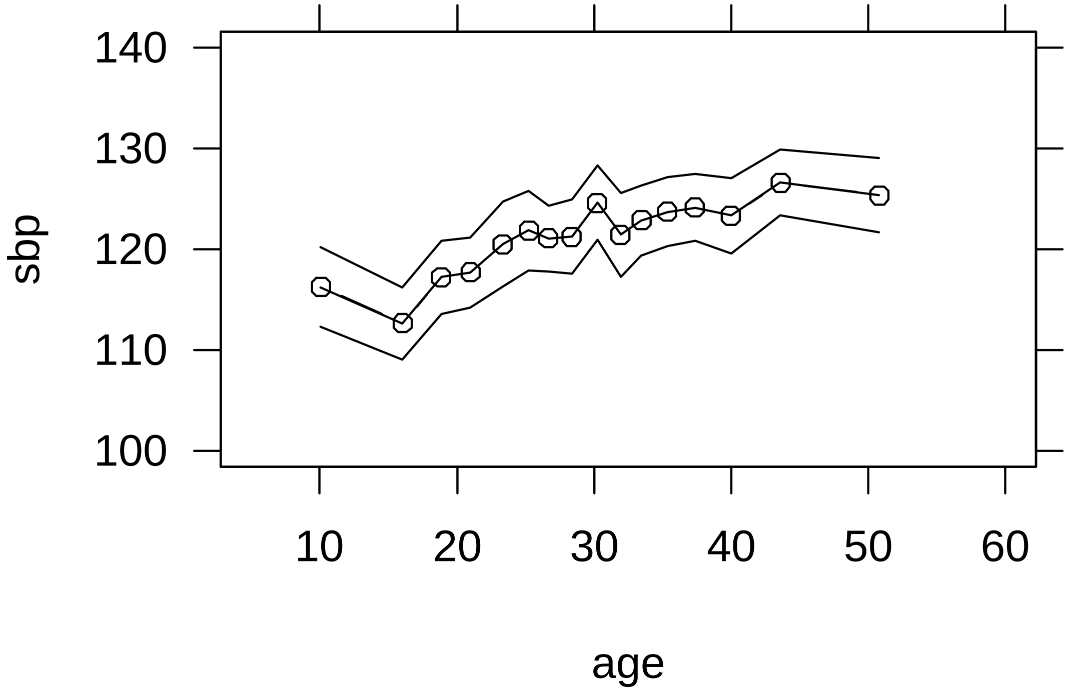


Figure 4.6: *Mean and parametric 0.95 confidence limits for means, for intervals of* `age` *containing 60 observations*

- Example: Display proportions and approximate 0.95 confidence limits from already-tabulated data

```
d ← expand.grid(continent=c('USA','Europe'),
                year=1999:2001)
d$proportion ← c(.2, .18, .19, .22, .23, .20)
d$SE ← c(.02, .01, .02, .015, .021, .025)

d
  continent year proportion    SE
1       USA 1999       0.20 0.020
2    Europe 1999       0.18 0.010
3       USA 2000       0.19 0.020
4    Europe 2000       0.22 0.015
5       USA 2001       0.23 0.021
6    Europe 2001       0.20 0.025

Dotplot(year ∼ Cbind(proportion, proportion-1.96*SE,
                     proportion+1.96*SE) |
        continent,  data=d, ylab='Year') # Figure 4.7
```

- To re-arrange the order of the vertical groups, use the `reorder.factor` function

- First just reverse the order of years on the y-axis

```
yr ← factor(d$year, 2001:1999)
Dotplot(yr ∼ Cbind(proportion, proportion-1.96*SE,
                   proportion+1.96*SE) |
        continent, data=d, ylab='Year')
```

- Next, reorder years by the average proportion over

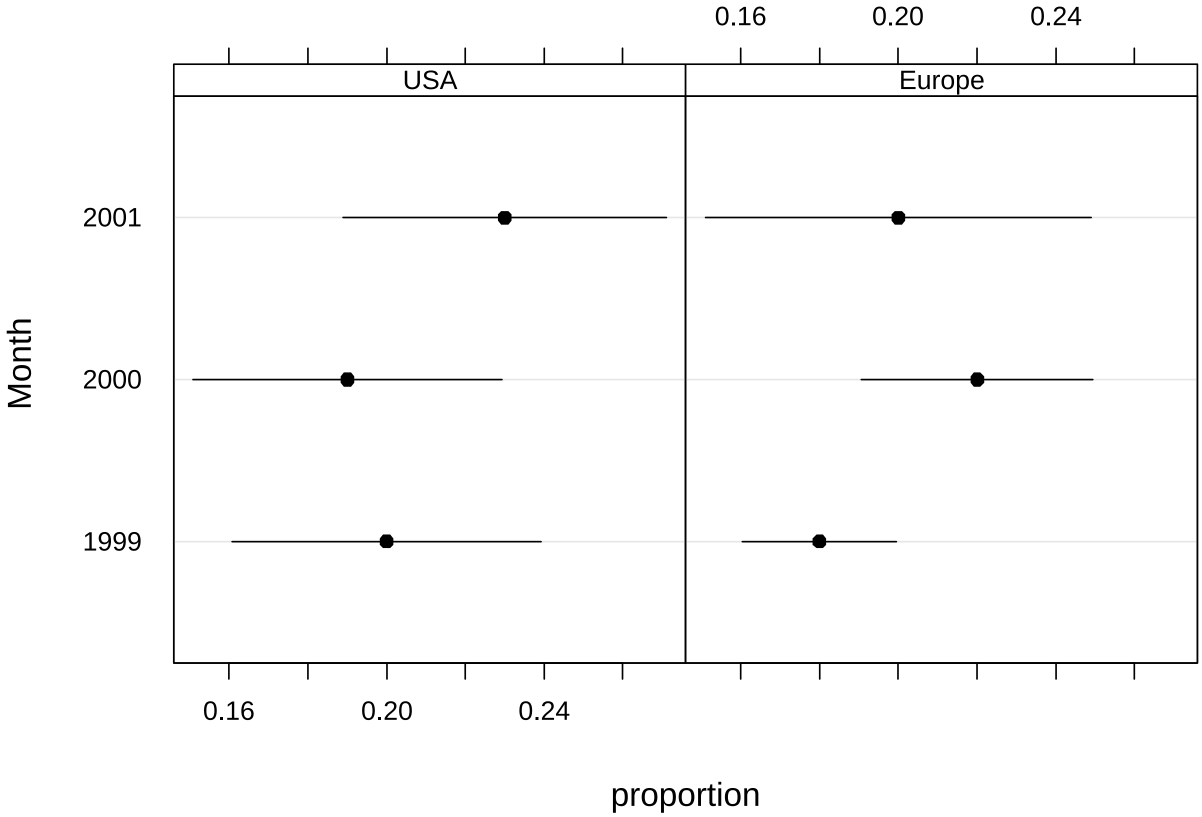


Figure 4.7: *Dot plot showing proportions and approximate 0.95 confidence limits for population probabilities*

the two continents

```
yr ← factor(d$year) # reorder.factor only accepts factors
yr ← reorder.factor(yr, d$proportion, mean)
levels(yr)

[1] "1999" "2000" "2001"
```

This happens to be in the original order so the dot plot will be the same as Figure 4.7

- To use more accurate Wilson confidence intervals on raw data:

```
set.seed(3)
d ← expand.grid(continent=c('USA','Europe'),
                 year=1999:2001, reps=1:100)
# Generate binary events from a population probability of
# 0.2 of the event, same for all years and continents
d$y ← ifelse(runif(6*100) ≤ .2, 1, 0)
rm(y)      # remove old y; don't confuse the following
attach(d)
s ← summarize(y, llist(continent,year),
               function(y) {
                n ← sum(!is.na(y))
                s ← sum(y, na.rm=T)
                binconf(s, n)
               }, type='matrix')
Dotplot(year ~ Cbind(y) | continent,
        data=s, ylab='Year')
# Same format of output as Figure 4.7
```

- Example: `dfr` data frame and associated raw response variable `y` from above

- Display a 5-number (5-quantile) summary (2 intervals, dot=median)

```
s ← summarize(y, llist(month,year), quantile,
                probs=c(.5,.05,.25,.75,.95), type='matrix')
Dotplot(month ∼ Cbind(y) | factor(year),
        data=s, ylab='Month')  # Figure 4.8
dev.off()
```

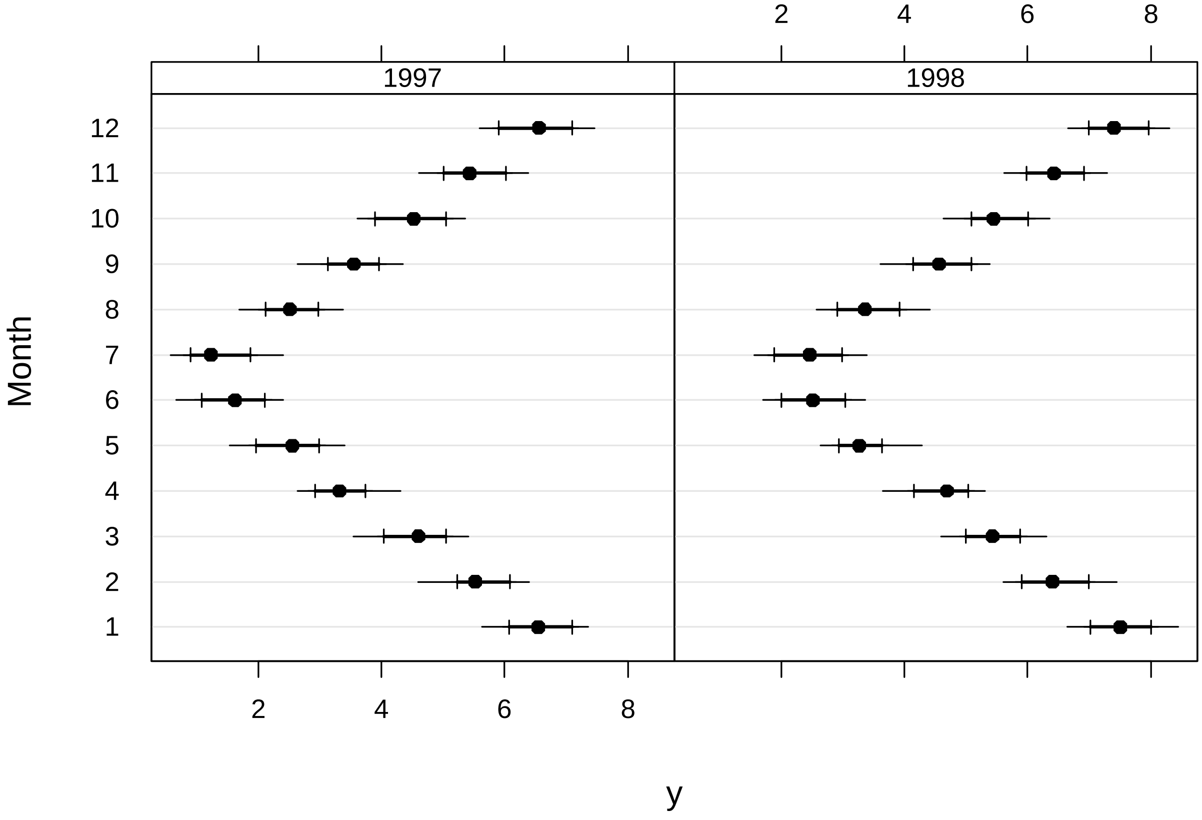


Figure 4.8: *Multi-tiered dot plot showing* $.05, .25, .5, .75, .95$ *quantiles of raw data*

4.7.9 Summary of Functions for Aggregating Data for Plotting

`tapply`

- Stratifies a single variable by one or a list of stratification variables
- When stratify by > 1 variable, result is a matrix which difficult to plot directly
- Hmisc `reShape` function can be used to re–shape the result into a data frame for plotting
- When stratify by a single variable, `tapply` creates a vector of summary statistics suitable for making a simple dot or bar plot without conditioning

`aggregate`

- Input = vector or a data frame and a `by` list of one or more stratification variables
- Handy to enclose the `by` variables in the `llist` function
- Can summarize many variables at once but only a single number such as the mean is computed for

each one

- `aggregate` does not preserve numeric stratification variables — it transforms them into factors which are not suitable for certain graphics
- Result is data frame

`summary.formula`

- Can compute separate summaries for each of the stratification variables
- Can also do × classifications when `method='cross'`
- Can summarize response variable using multiple statistics (e.g., mean and median)
- If specify a `fun` function that can deal specially with matrices, you can summarize multiple–column response variables
- Creates special objects and has special methods for presenting them
  - `print` method for printing a table in ASCII text

  format

  - `plot` method for plotting the result (not available for `method='cross'`
  - LaTeX method for typesetting the table, allowing the use of multiple fonts, character sizes, subscripts, superscripts, bold, etc.

- Don't plot the results of `summary.formula` using one of the Trellis/Lattice functions.

`summarize`

- Similar purpose as `aggregate` but with some differences
- Will summarize only a single response variable but the `FUN` function can summarize it with many statistics
- Can compute multiple quantiles or upper and lower limits for error bars
- Will not convert numeric stratifiers to factors, so

output is suitable for summarizing data for `xyplot` or `xYplot` when the stratification variable needs to be on the x–axis

- Only does cross–classification
- Creates an ordinary data frame suitable for any use in S, especially for passing as a `data` argument to Trellis/Lattice graphics functions
- Can also easily use the GUI to graph this data frame

`method=function` with `xYplot`: Automatically aggregates data to be plotted when central tendency and upper and lower bands are of interest.

Chapter 5

Nonparametric Trend Lines

- Continuous X, continuous or binary Y
- Nonparametric smoother only assumes that the shape of the relationship between X and Y is smooth
- A smoother is like a moving average but better
  - Moving average is a moving flat line approximation
  - Moving averages have problems in the left and right tails
- Best all-purpose smoother: `loess`

- Is called a scatterplot smoother or moving weighted linear regression

- By having moving slope and intercept, with overlapping windows, the smooth curve is more accurate and has no problems in left and right tails

- `loess` can handle binary response variable if you turn off outlier rejection (i.e., tell the algorithm to do no extra iterations)

- Basic R function for `loess` smoothing is `lowess`:

```
plot(age, sysbp)
lines(lowess(age, sysbp))
```

- To use more than two variables use the function called `loess` which uses the statistical formula language

- Hmisc `plsmo` function plots `loess` or “super smoother” (`supsmu`) estimates with several options including automatic stratification on a discrete variable

```
plsmo(age, sysbp, group=sex, datadensity=T)
# 2 curves with rug plots
```

- Example using `titanic3` dataset from Web site

```
attach(titanic3)
plsmo(age, survived, group=interaction(pclass,sex),
      datadensity=T)   # Figure 5.1
dev.off()
```

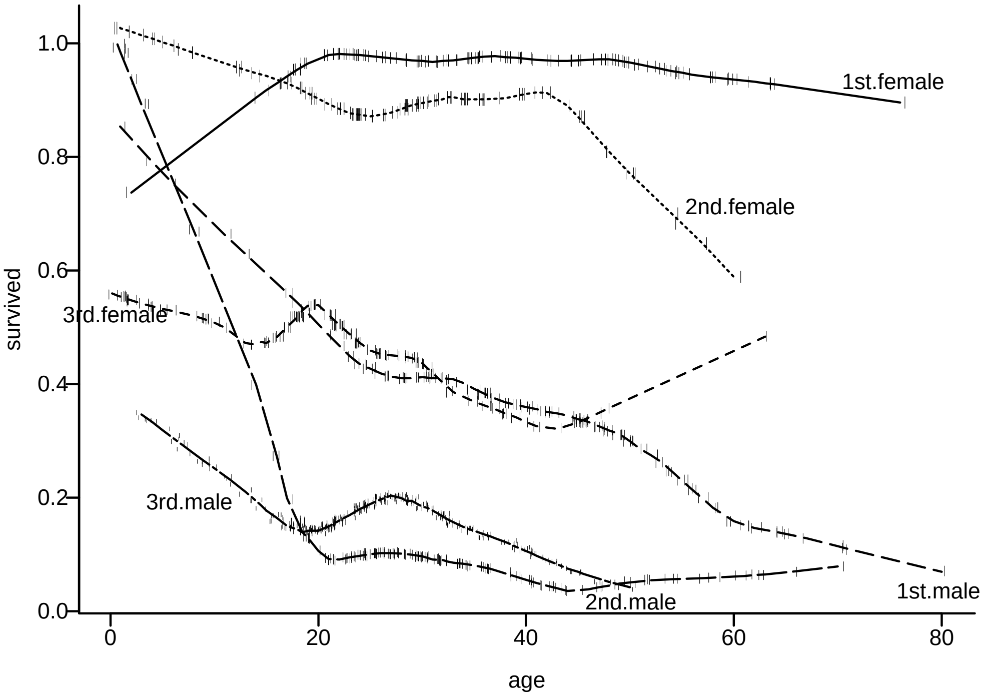


Figure 5.1: `loess` *smoothed estimates of the probability of surviving the* Titanic *as a function of passenger age, sex, and ticket class*

`interaction(a,b)` creates a new `factor` variable containing the cross-classifications of the two constituent variables

- `plsmo` automatically turns outlier rejection off if the `y` variables has only two unique values

- `plsmo` automatically labels curves by levels of the `group` variable[a]

- You can use `plsmo` as a `panel` function to `xyplot`:

```
xyplot(sysbp ~ age | race, groups=sex, panel=panel.plsmo)
```

- Other ways to get trend lines using Trellis/Lattice are given in Section 4.7.3

---

[a] Note that `group` is **not** plural, which is inconsistent with the Trellis/Lattice `groups` variable used for superposition.

Chapter 6

A Newer Graphical Model in R: `ggplot2`

The `ggplot2` package by Hadley Wickham, RStudio, was developed several years after the `lattice` package in R. It is based on Wilkinson's *Grammar of Graphics*. Dr Wickham provided the following `ggplot2` examples and text.

6.1 Basics

This example shows some of the basic features of `ggplot2`, focusing particularly on the scatterplot, one of the most important statistical graphics.

There are two ways of specifying `ggplot2` graphics: a concise way that uses some "magic" to figure out what you want, and a more verbose, but explicit form. These examples use the more explicit form (so it's easier to see what's happening), but for everyday use you can use `qplot`, which mimics the base `plot` function, and can save a lot of typing.

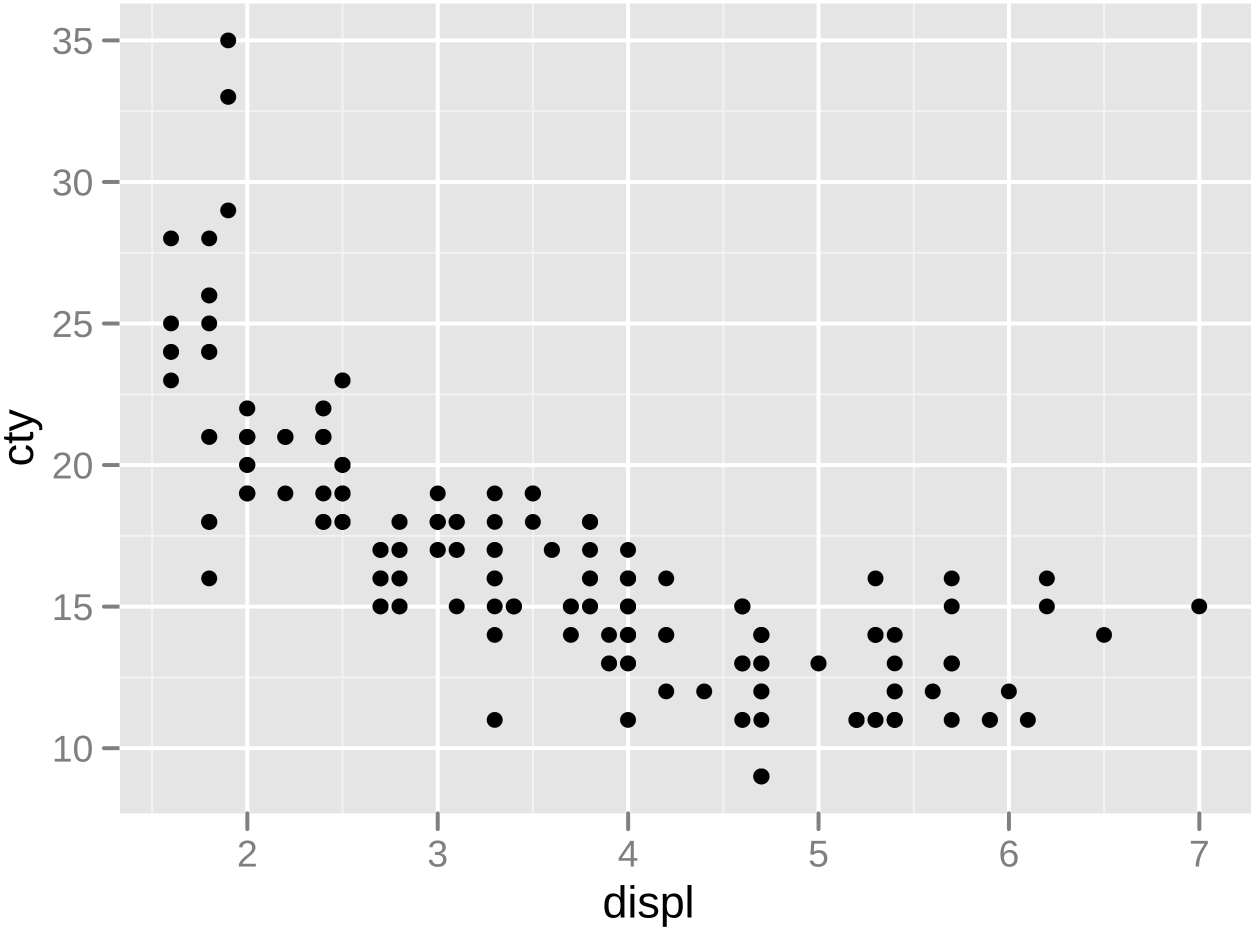
```
require(ggplot2)
require(plyr)
mpg ← subset(mpg, cyl != 5)
```

To create a `ggplot2` graphic you start off with the `ggplot2` function. This function specifies the default dataset and default mapping from the data to things that you can see on the plot, the aesthetics.

The following call uses the `mpg` dataset (a set of economy related variables for 235 cars, measured in 1999 and 2008). The aesthetic mapping says we want to map the x-position to displacement and the y-position to mpg in the city. We then add a layer of points with `geom_point()`. The geom, short for geometric object, determines the basic type of plot. With *points* we get a scatterplot.

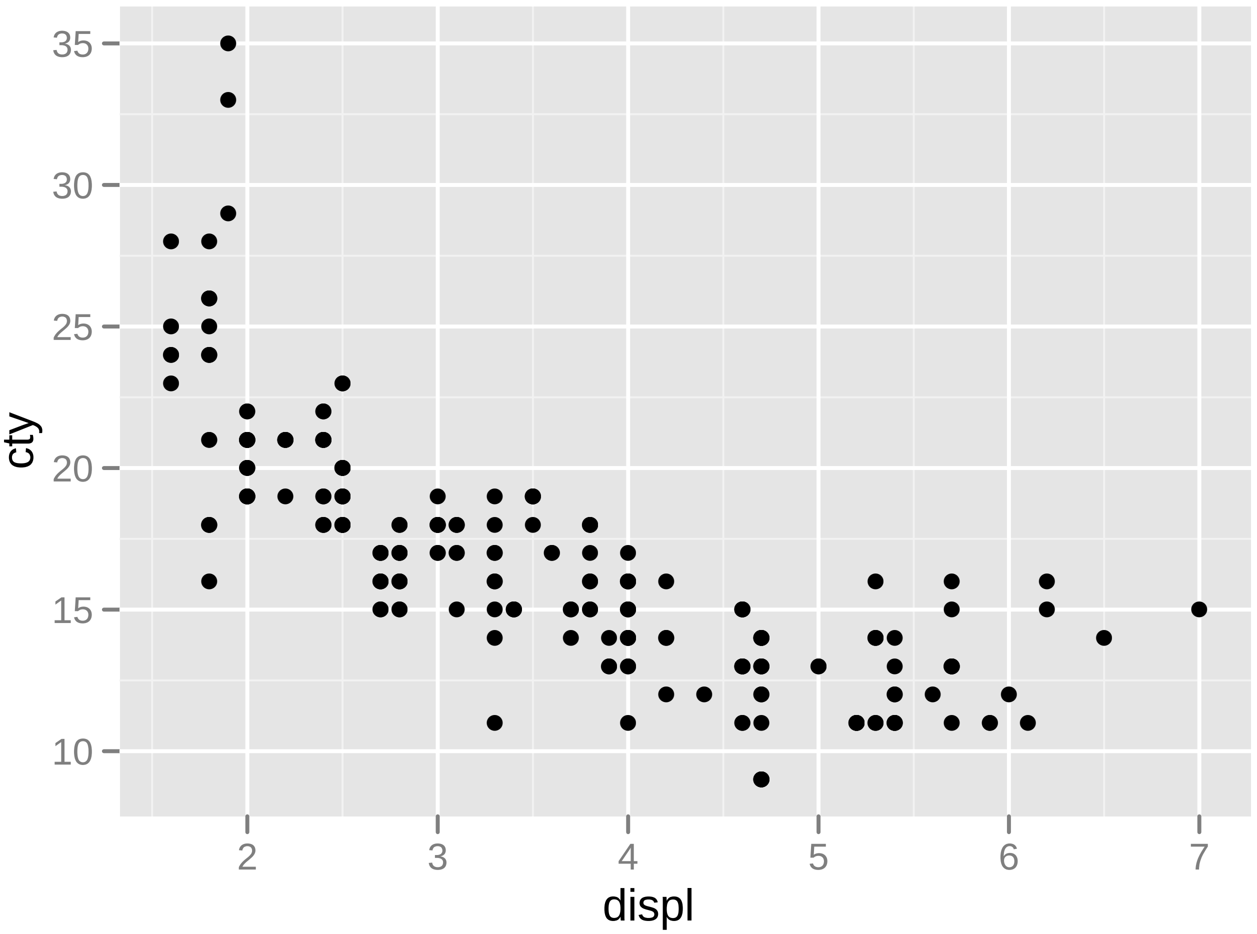
Note: Like `lattice`, `ggplot2` plotting commands must be enclosed in `print()` to render the graphic unless running R interactively. Here `Sweave` is used so `print()` is required.

```
print(ggplot(mpg, aes(x = displ, y = cty)) +
      geom_point())
```



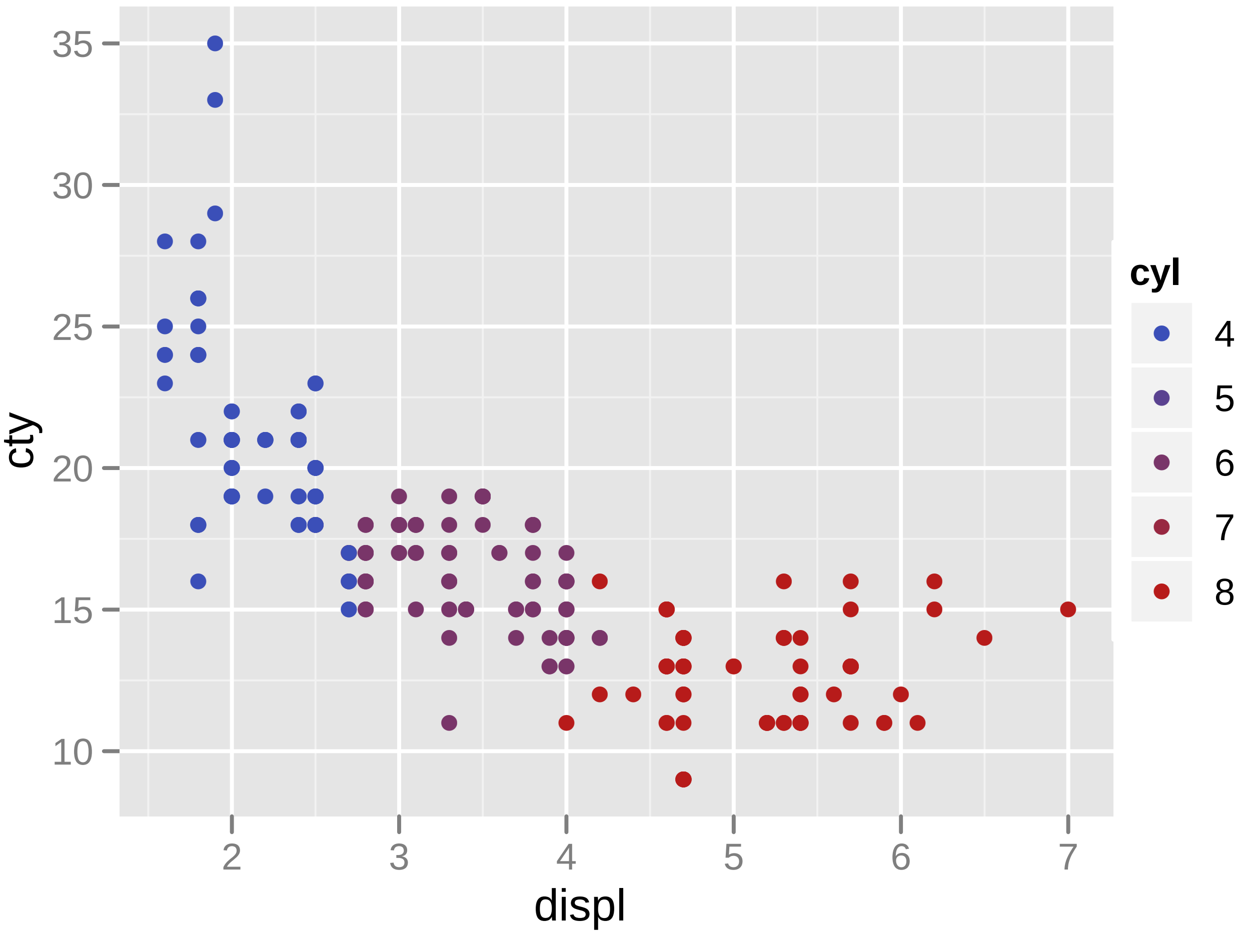
Because almost every plot uses both x and y positions, they are the first two arguments to `aes`, and we can omit their names. We'll do that from here on.

```
print(ggplot(mpg, aes(displ, cty)) +
      geom_point())
```



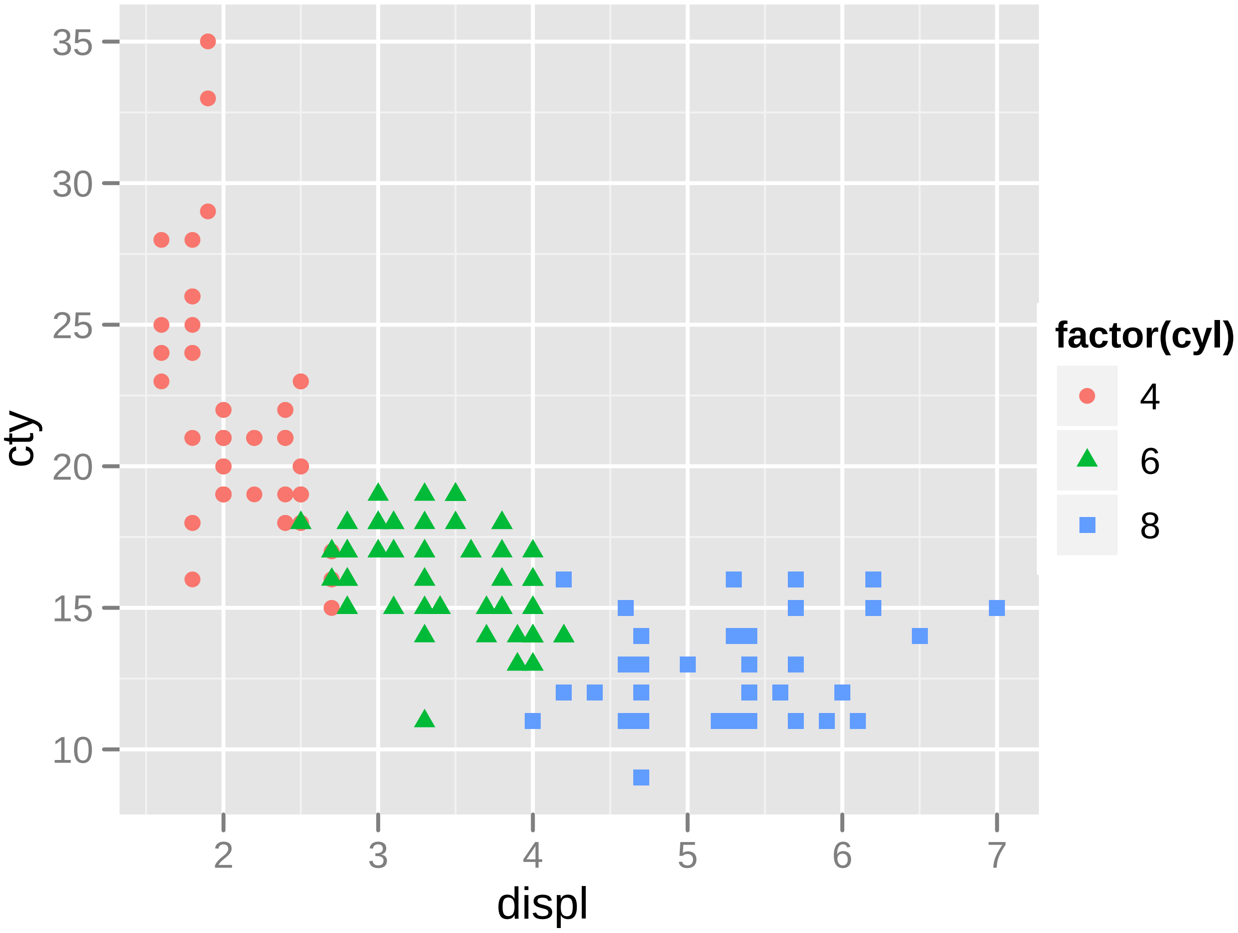
We can also map variables to other aesthetics like colour, size and shape. `ggplot2` will automatically pick a sensible scale (here a continuous colour gradient) and draw a legend for us.

```
print(ggplot(mpg, aes(displ, cty, colour = cyl)) +
        geom_point())
```



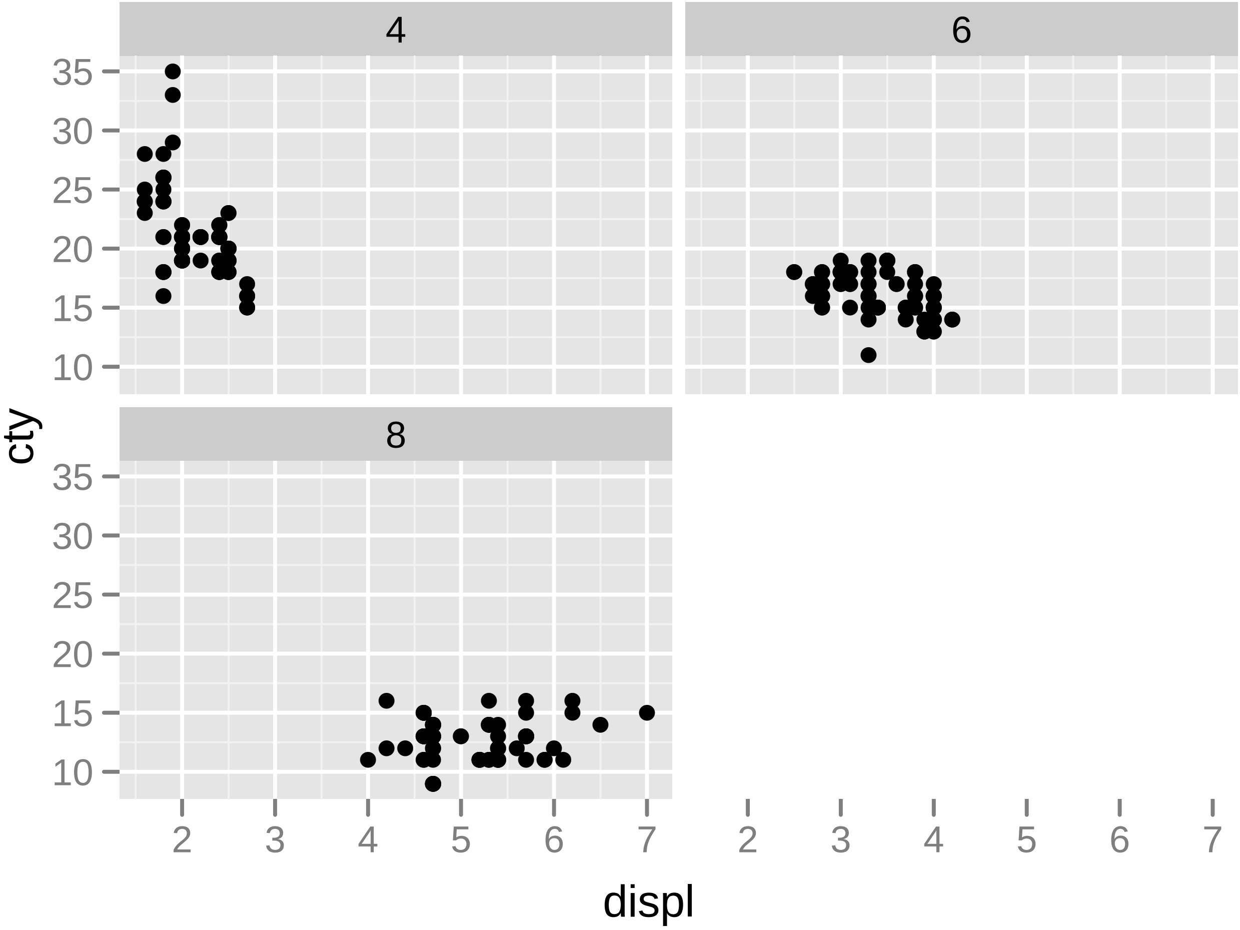
If we use a categorical variable we get a different type of scale. We can also map the same variable to multiple aesthetics, and `ggplot2` knows what to do. Notice how both colour and shape vary in the the legend.

```
print(ggplot(mpg, aes(displ, cty, colour = factor(cyl),
                       shape = factor(cyl))) +
        geom_point())
```



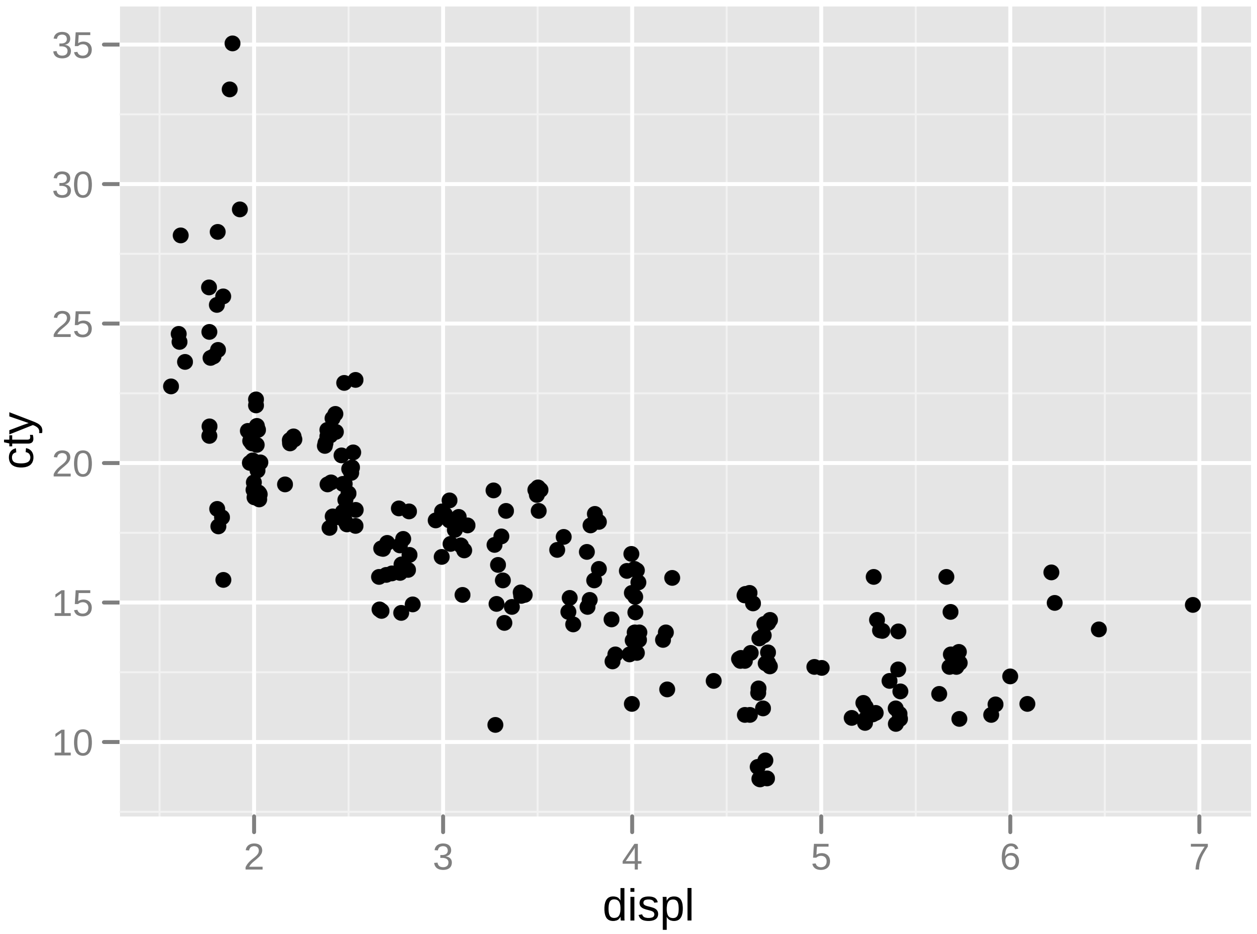
Another powerful technique for visualisation is facetting (also known as conditioning or trellising). This makes a separate panel for each subset of the data:

```
print(ggplot(mpg, aes(displ, cty)) +
      geom_point() +
      facet_wrap(~ cyl))
```



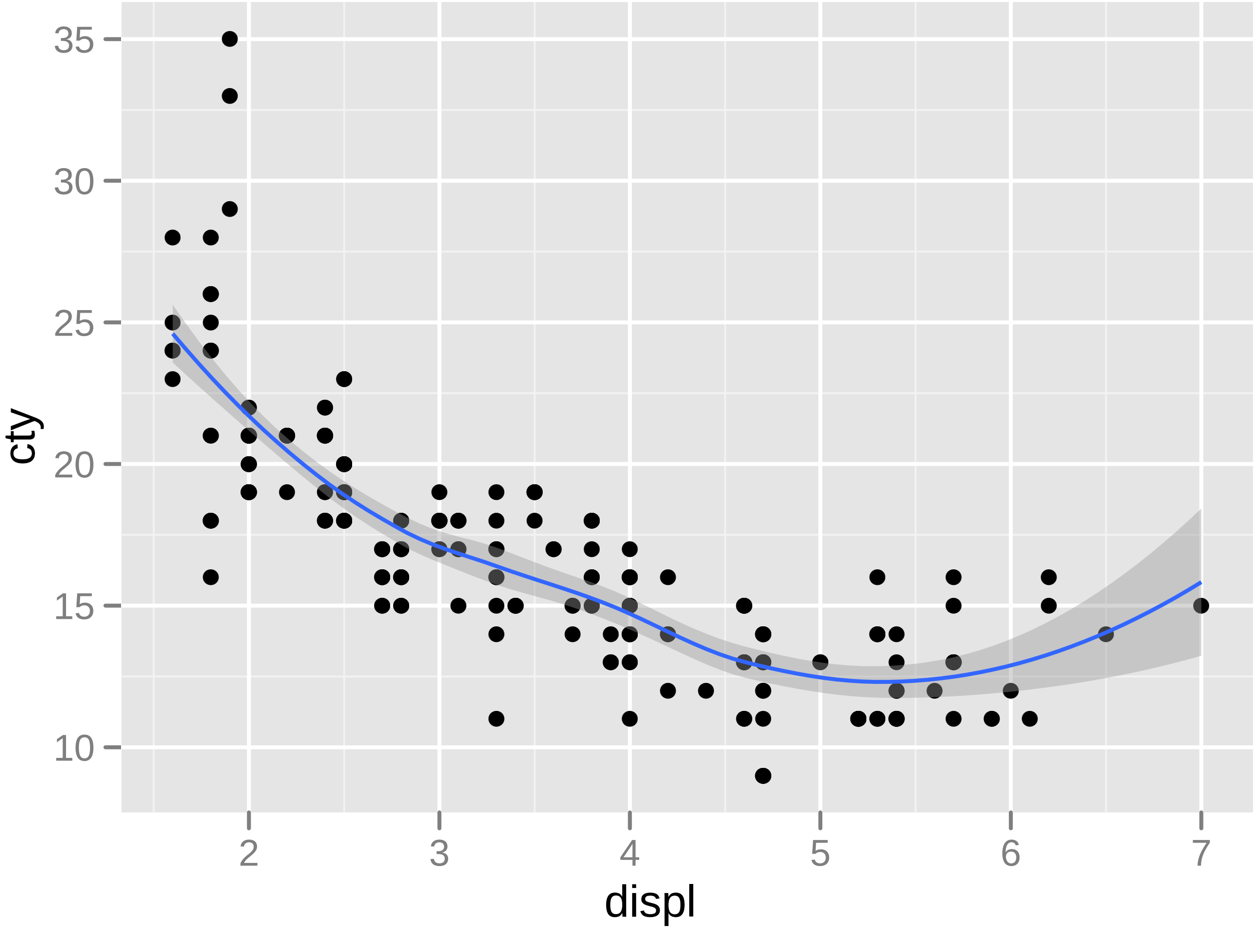
There are plenty of other geoms we could use with this data. For example, we could use the `jitter` geom, which jitters each point by a small amount:

```
print(ggplot(mpg, aes(displ, cty)) +
      geom_jitter())
```



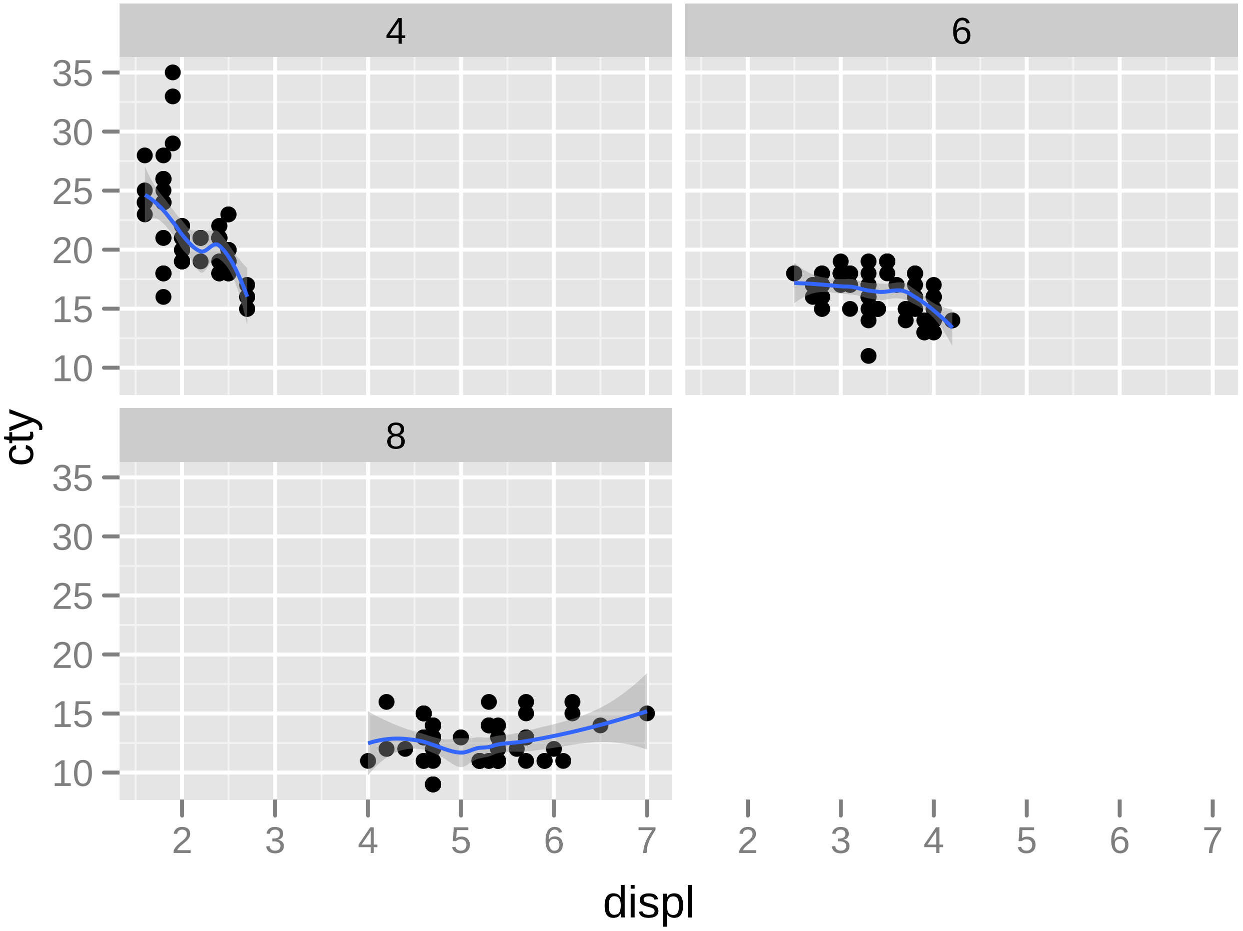
Or we could add a smooth curve

```
print(ggplot(mpg, aes(displ, cty)) +
      geom_point() +
      geom_smooth())
```

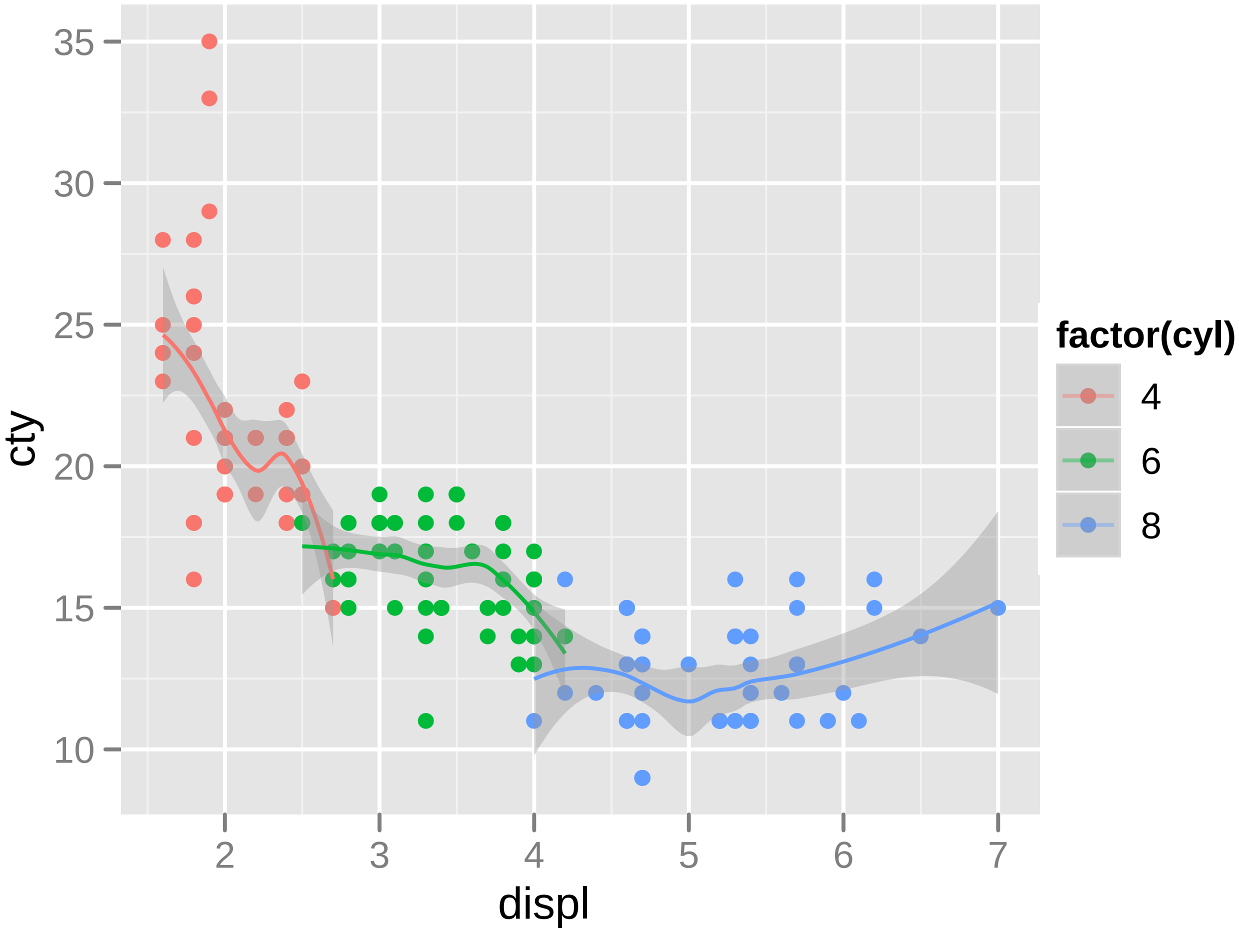


Because we are specifying what we want in the graphic, instead of constructing it piece-by-piece, `ggplot2` knows what to do if we add colour or facetting to a plot with a smooth line on it:

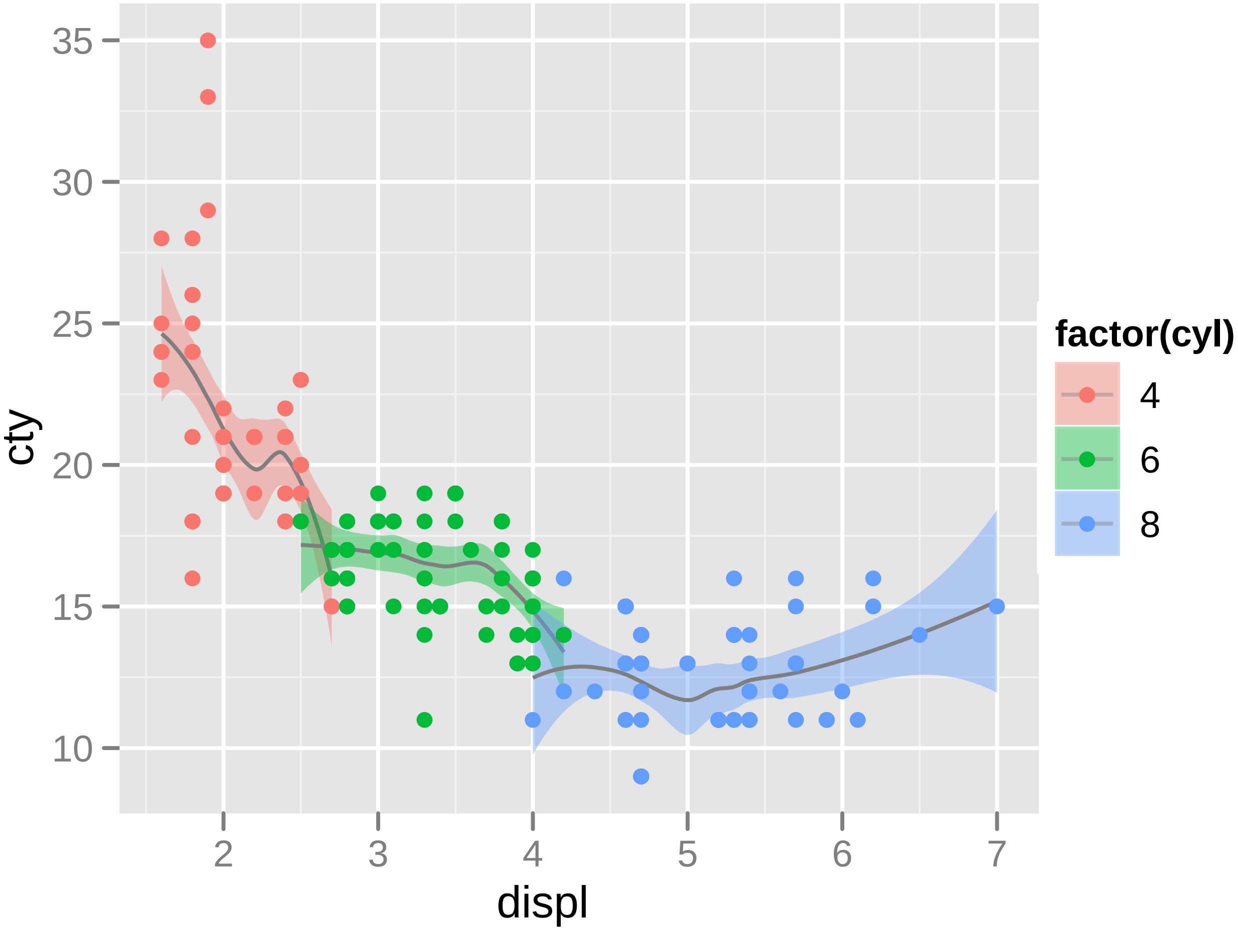
```
print(ggplot(mpg, aes(displ, cty)) +
      geom_point() +
      geom_smooth() +
      facet_wrap(~ cyl))
```



```
print(ggplot(mpg, aes(displ, cty, colour = factor(cyl))) +
        geom_point() +
        geom_smooth())
```



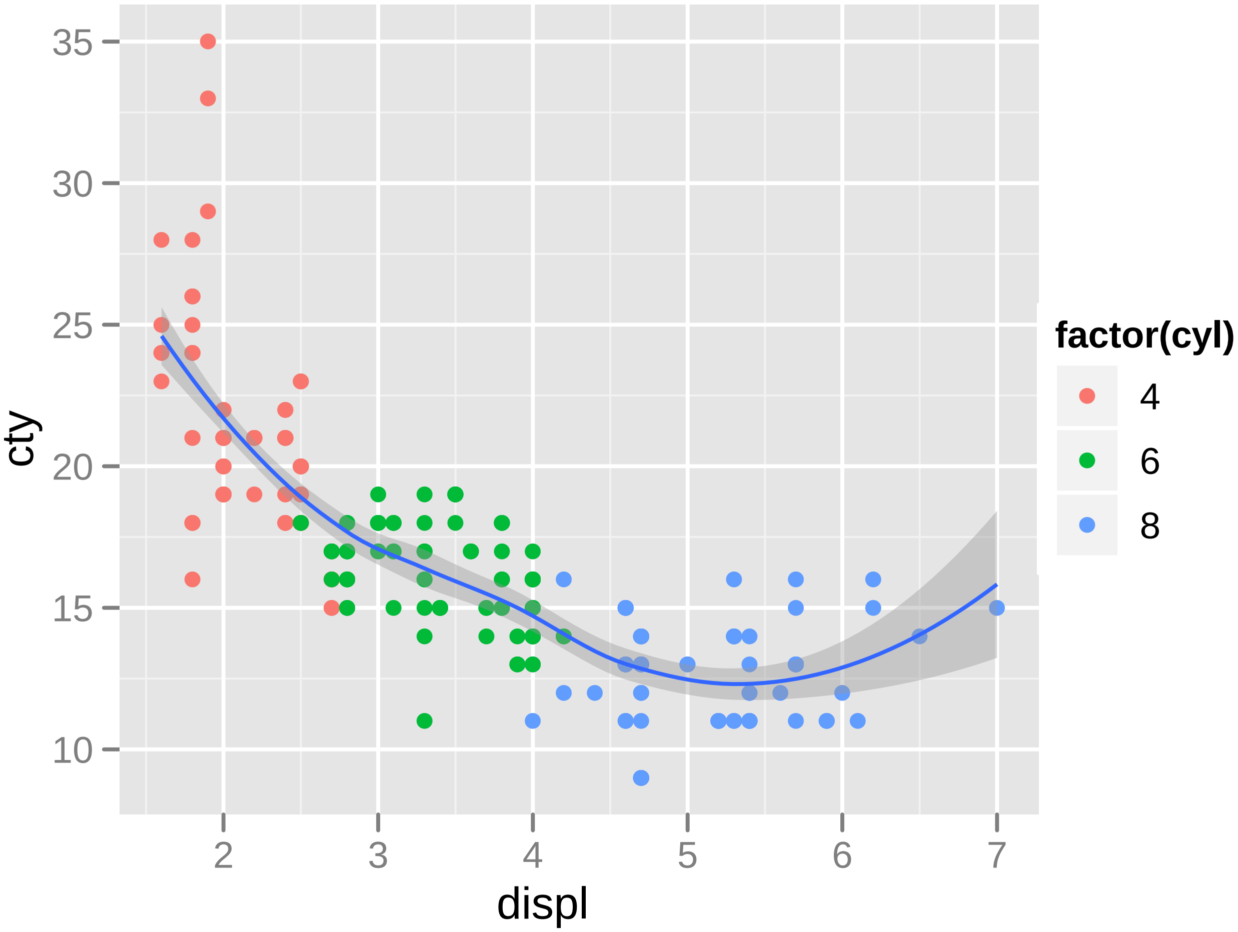
```
print(ggplot(mpg, aes(displ, cty, colour = factor(cyl),
                      fill = factor(cyl))) +
      geom_smooth(colour = "grey50") +
      geom_point())
```



Note how the legend does its best to capture what's on the plot.

Each call to `geom_xxx` adds a new layer to the plot. Different layers can use different aesthetic mappings or even different datasets, making it possible to create rich, sophisticated graphics. In the following simple example we move the colour specification out of the defaults (which apply to all layers), into the point layer. Notice how we just get one smooth line.

```
print(ggplot(mpg, aes(displ, cty)) +
      geom_point(aes(colour = factor(cyl))) +
      geom_smooth())
```



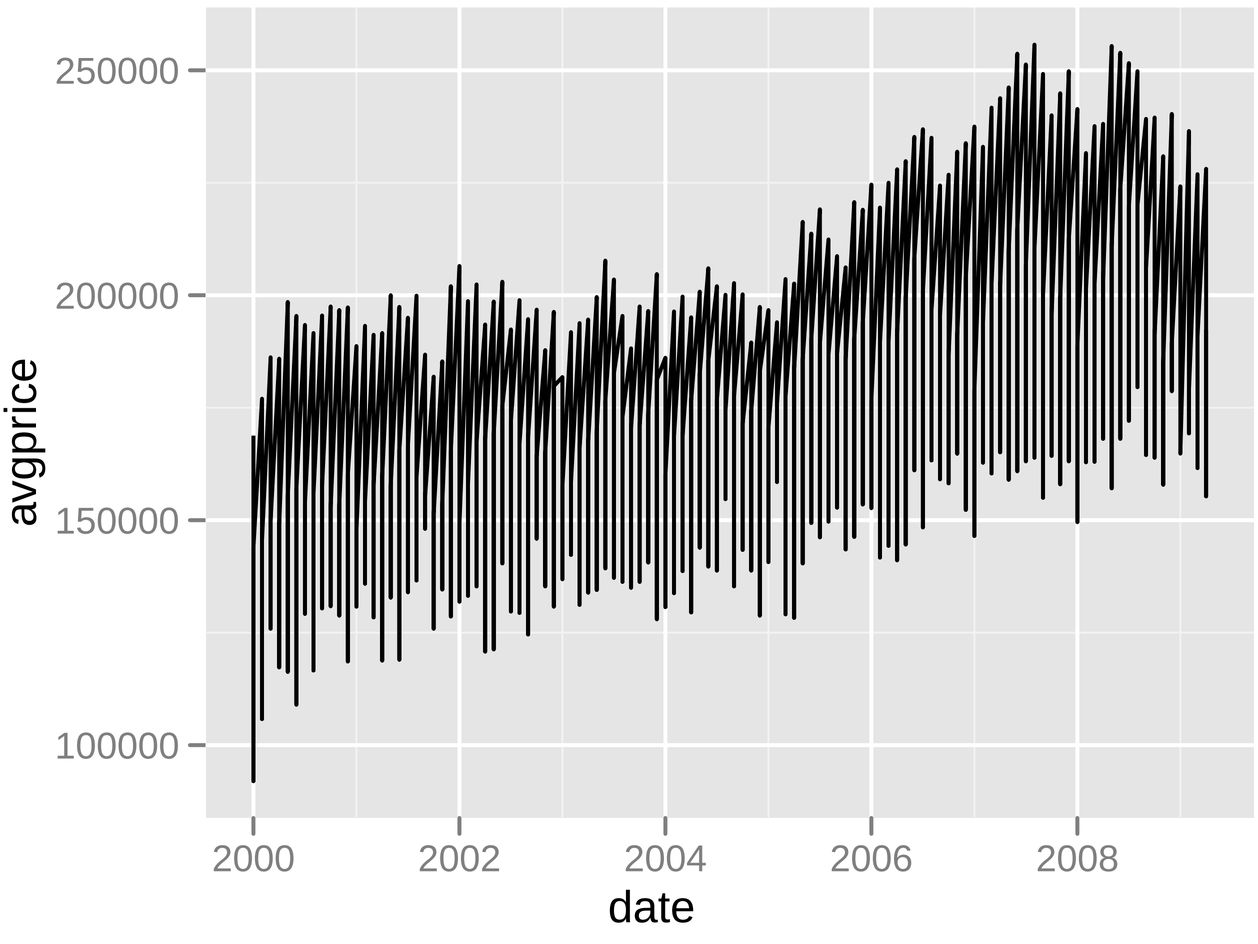
This quick introduction has only scratched the surface of `ggplot2`. To learn more, see `http://had.co.nz/ggplot2`.

6.2 Time Series Examples

This is a graphical exploration of time series of monthly housing sales in three interesting cities in Texas from 2000-2009. Houston is big, Austin medium sized, and College Station small, with a high proportion (55,000 / 80,000) of college students.

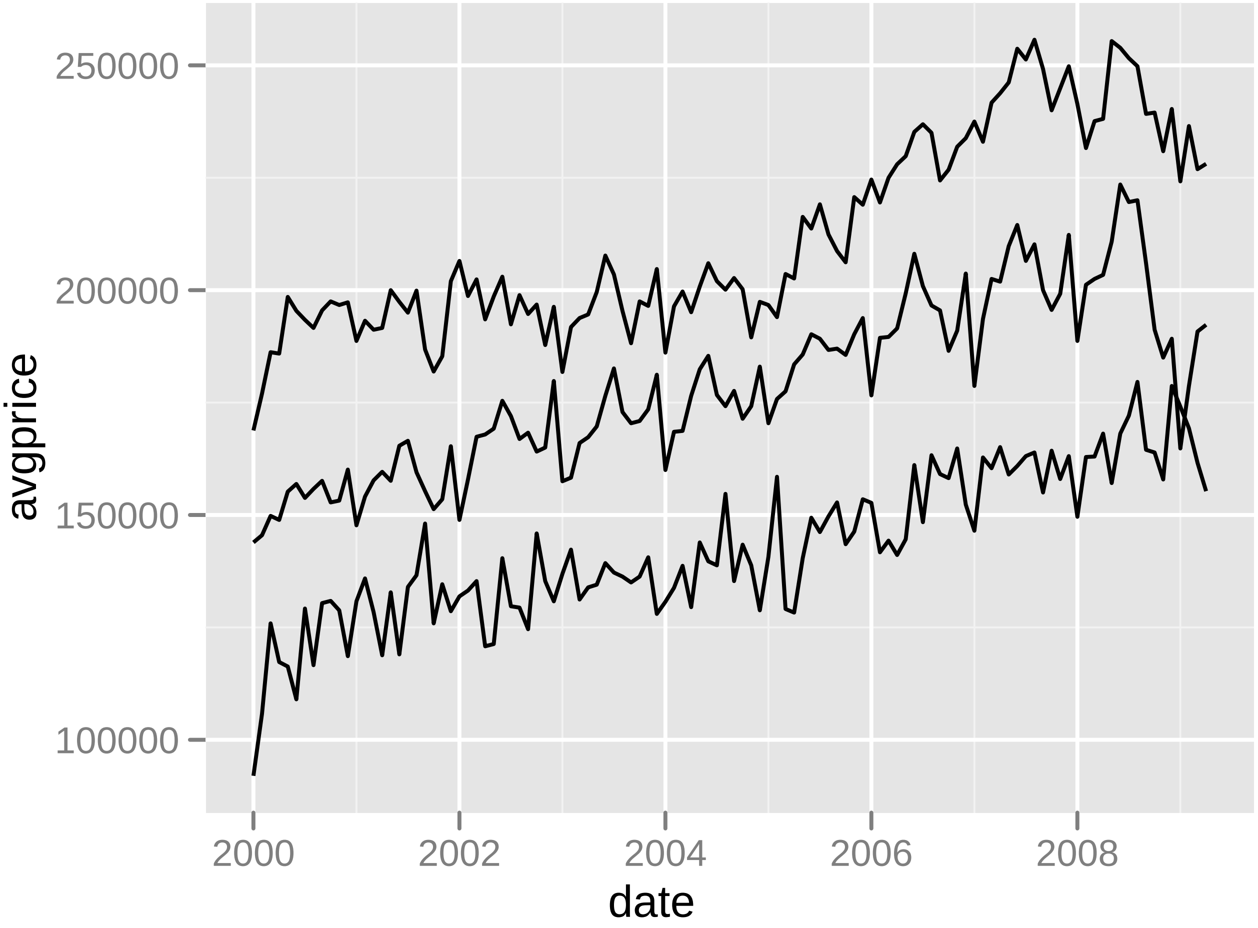
```
load('houseSales.rda')
tx ← subset(houseSales,
             city %in% c("Austin", "Houston", "Bryan-College Station"))
```

```
print(ggplot(tx, aes(date, avgprice)) + geom_line())
```



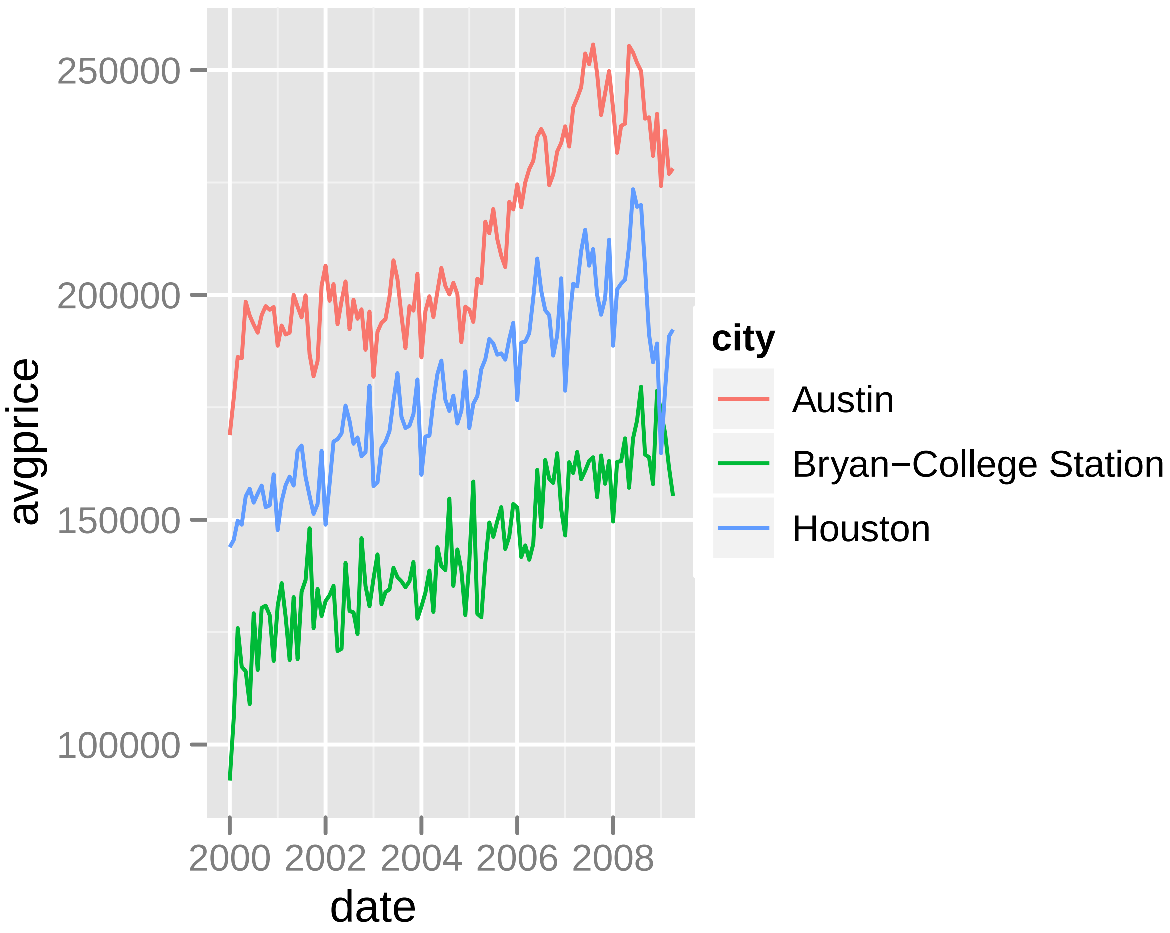
No good - need to specify which variable splits the data up into lines.

```
print(ggplot(tx, aes(date, avgprice, group = city)) + geom_line())
```



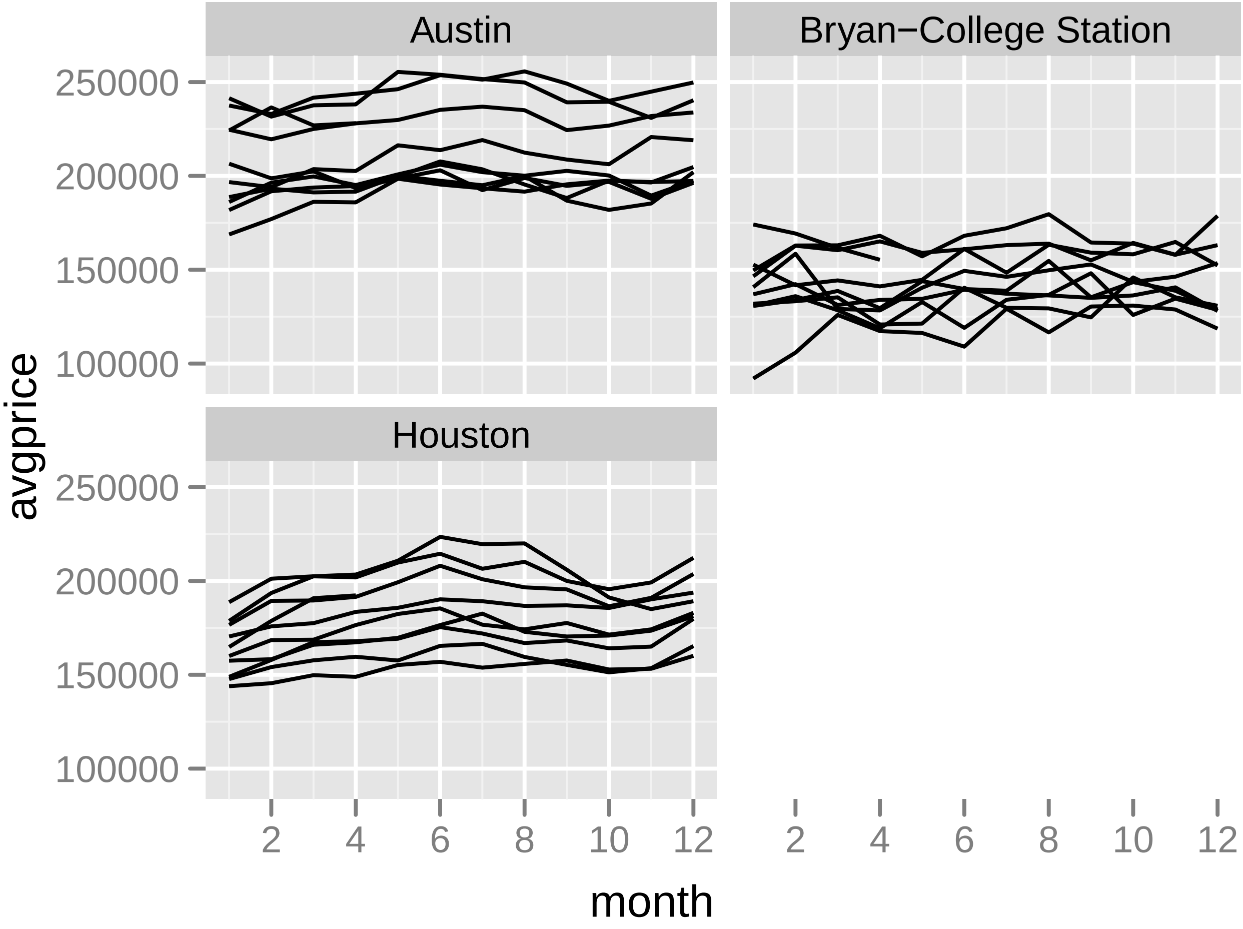
Alternatively we can specify an aesthetic and get a legend which tells us which line is which.

```
print(ggplot(tx, aes(date, avgprice, colour = city)) + geom_line())
```



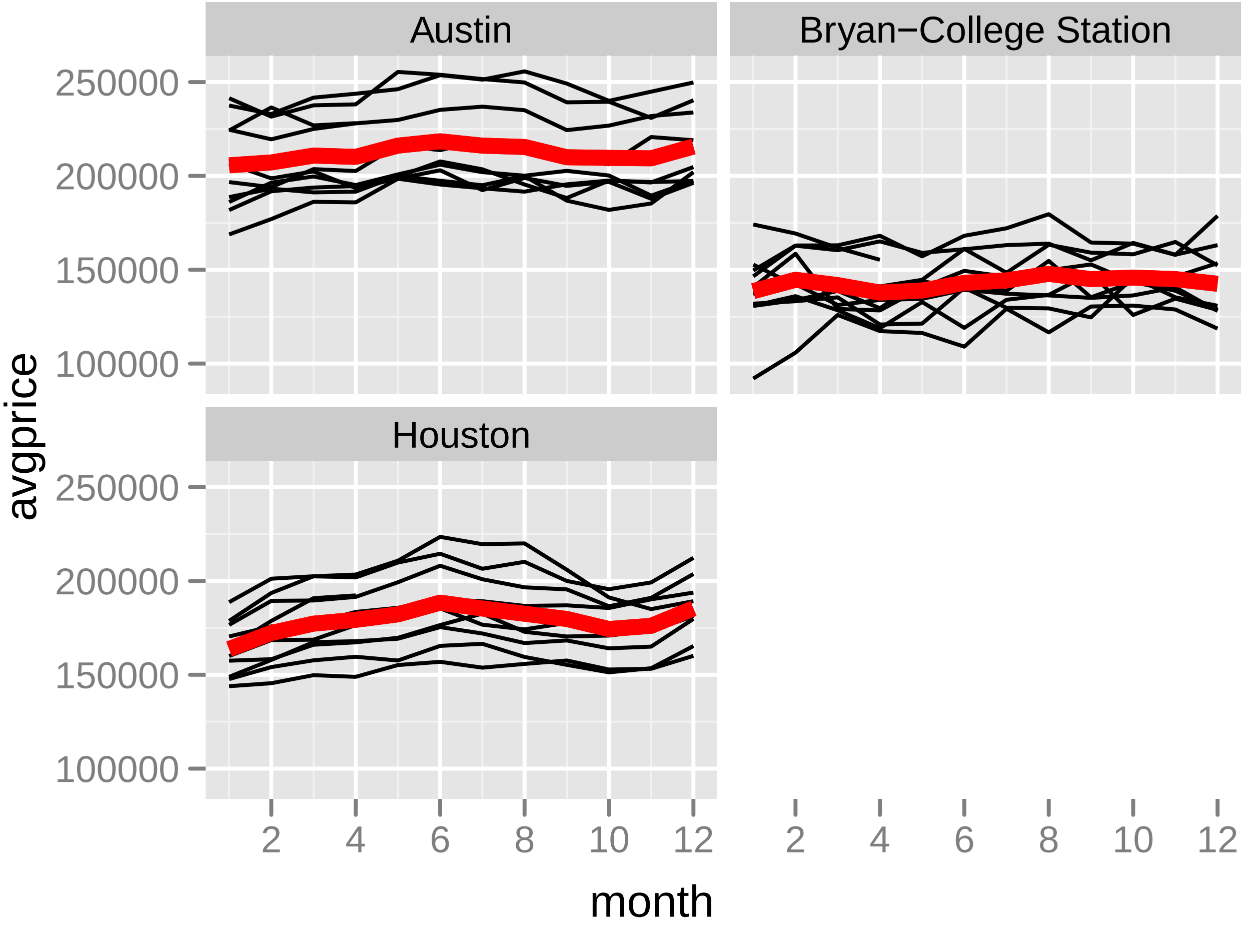
Is there a hint of a seasonal effect? Let's investigate further by drawing a time series plot with month on the x-axis and a line for each year. We'll also use facetting to divide up the cities.

```
print(ggplot(tx, aes(month, avgprice, group = year)) +
      geom_line() +
      facet_wrap(~ city))
```



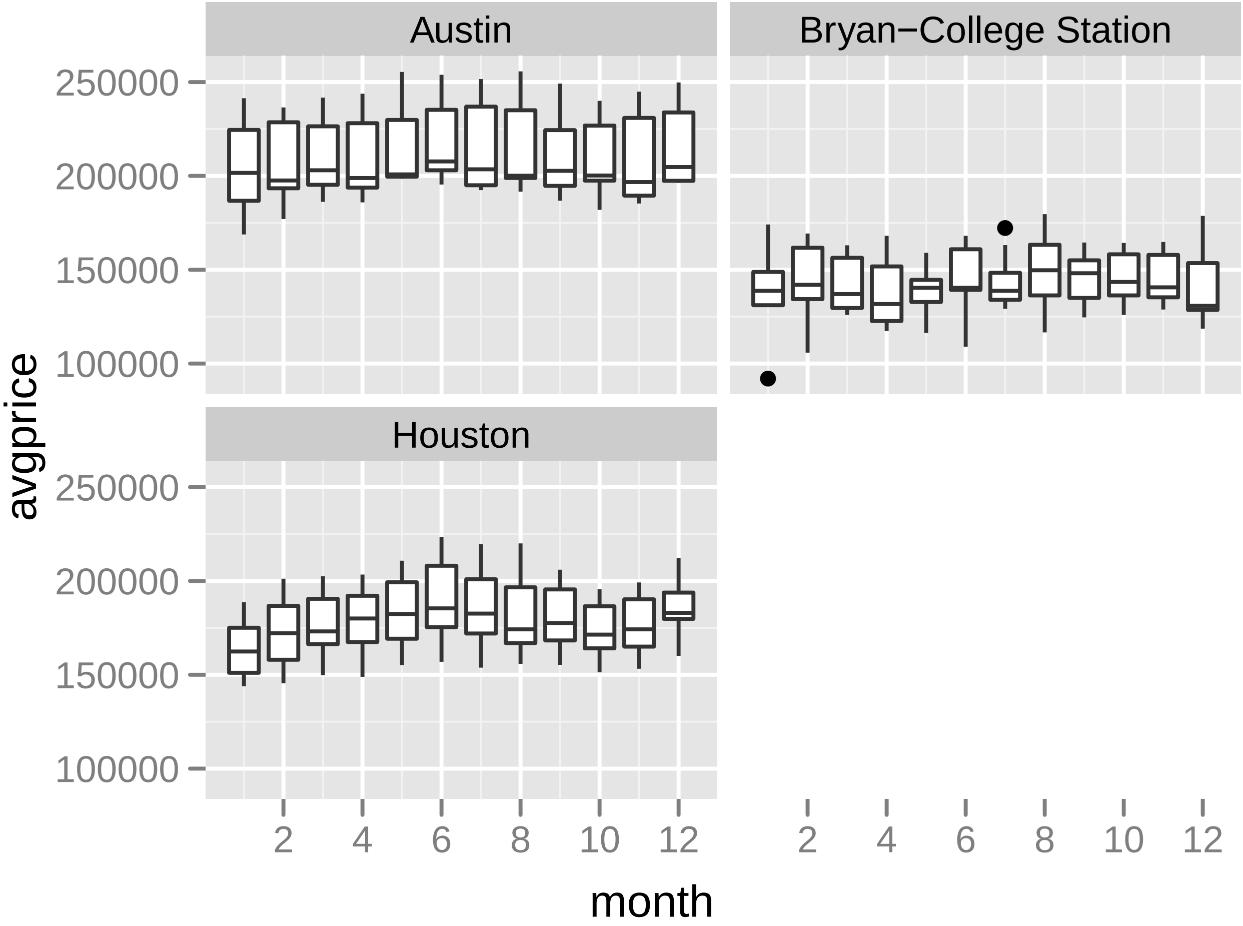
We can also add to the previous plot. Here we add a thick red line showing the mean price for each month. We need to use a different grouping because otherwise we'd taking the mean of a single observation

```
print(last_plot() +
      stat_summary(aes(group = 1), fun.y="mean", geom="line",
      colour ="red", size=2))
```



Alternatively, we could draw a boxplot for each month.

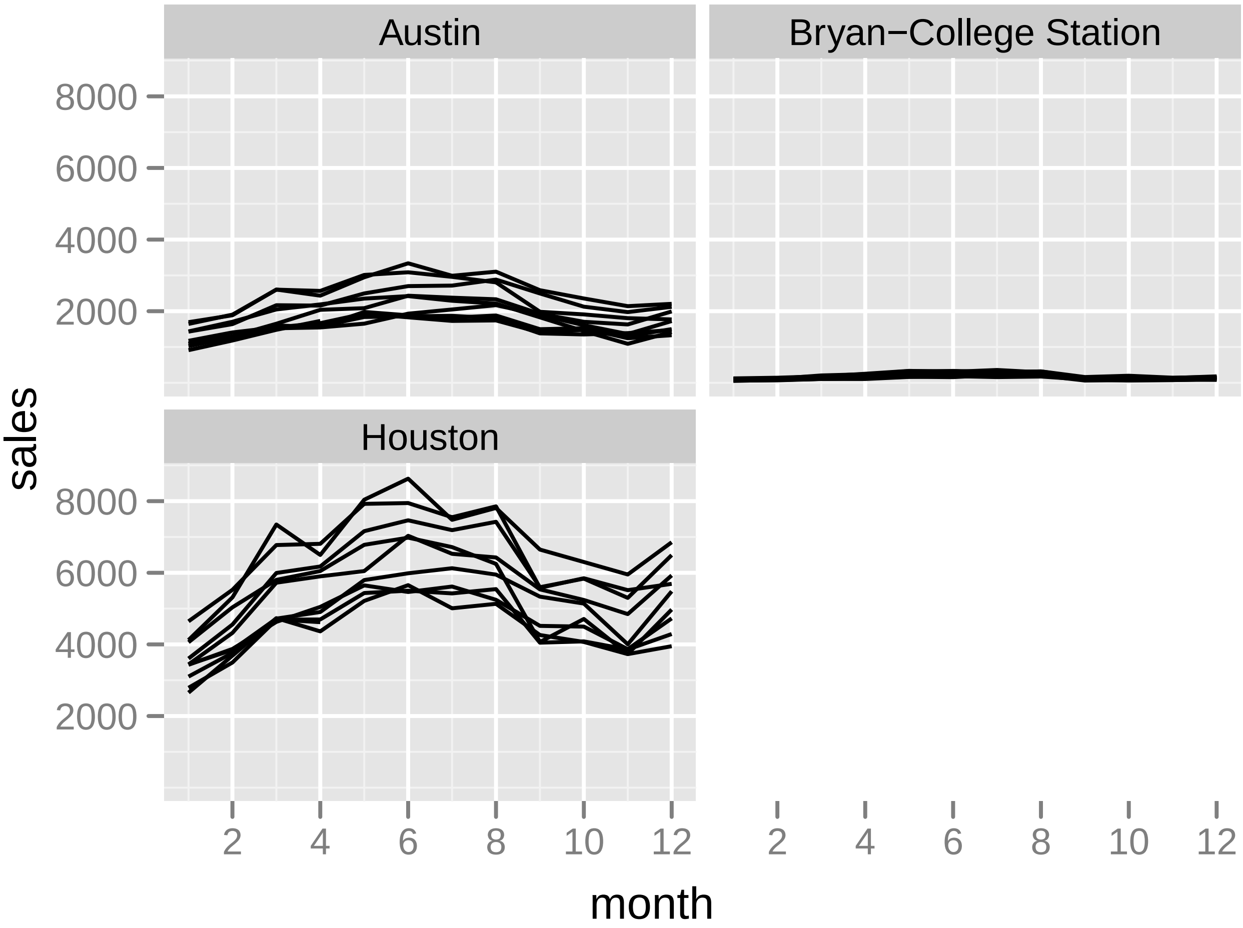
```
print(ggplot(tx, aes(month, avgprice, group = month)) +
      geom_boxplot() +
      facet_wrap(~ city))
```



Houston seems to have some seasonal differences, but it's hard to see any different in Austin or College Station. We'd probably want to check this with more formal inferential procedures

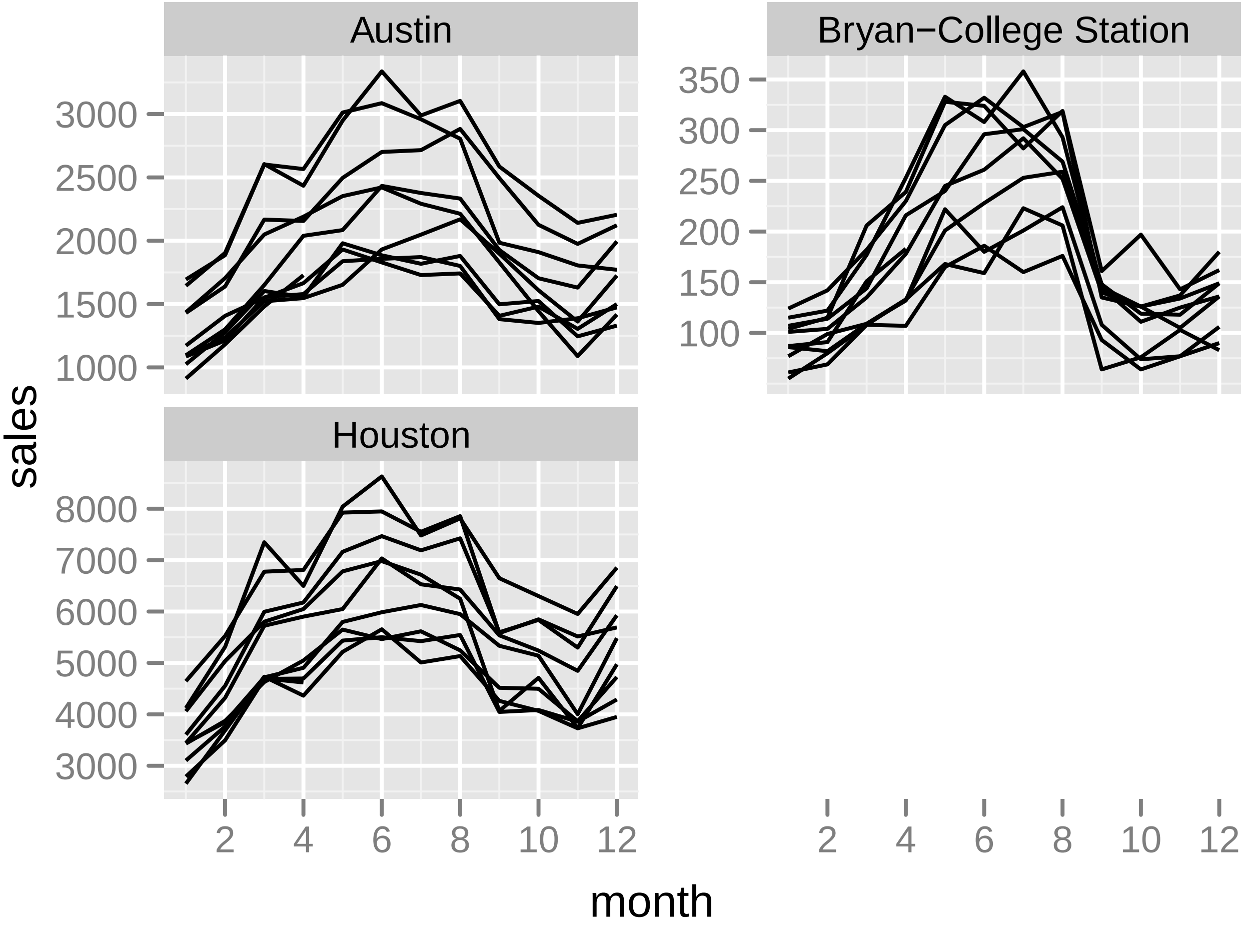
There's a much strong seasonal effect on sales:

```
print(ggplot(tx, aes(month, sales, group = year)) +
      geom_line() +
      facet_wrap(~ city))
```



But it's hard to compare the different cities because of large difference in total sales. With free scales each panel has it's own y scale, and you can see the seasonal pattern is a bit different in College Station. Maybe it's because it's a college town.

```
print(ggplot(tx, aes(month, sales, group = year)) +
      geom_line() +
      facet_wrap(~ city, scales = "free_y"))
```


Austin
Bryan−College Station
Houston
sales
month