A collection of some commonly used and some newly developed methods for the visualization of outcomes in oncology studies include Kaplan-Meier curves, forest plots, funnel plots, violin plots, waterfall plots, spider plots, swimmer plot, heatmaps, circos plots, transit map diagrams and network analysis diagrams. Previous articles in this blog presented an introduction to forest plots,Â violin plots and waterfall plots as well as provided some R code for the generation of these plots. As a continuation of the series, the current article provides an introduction to spider plots for the visualization of tumor response and generation of the same using R.

Waterfall Plot

In this post, we illustrate the creation of waterfall plots in R.

In a typical waterfall plot, the x-axis serves as the baseline value of the response variable. For each subject, vertical bars are drawn from the baseline, either in the positive or negative direction to depict the change from baseline in the response for the subject. The y-axis thus represents the change from baseline in the response, usually expressed as a percentage, for e.g., percent change in the size of the tumor or percent change in some marker level. Most importantly, in a waterfall plot, the bars are ordered in the decreasing order of the percent change values.

Though waterfall plots have gained popularity in oncology, they can be used for data visualization in other clinical trials as well, where the response is expressed as a change from baseline.

**Dataset:**

Instead of a tumor growth dataset, we illustrate creation of waterfall plots for the visual depiction of a quality of life data. A quality of life dataset, dataqol2 is available with the R package QoLR.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | require(QoLR)  ?dataqol2  data(dataqol2)  head(dataqol2)  dataqol2$id <- as.factor(dataqol2$id)  dataqol2$time <- as.factor(dataqol2$time)  dataqol2$arm <- as.factor(dataqol2$arm) |

dataqol2 contains longitudinal data on scores for 2 quality of life measures (QoL and pain) for 60 subjects. In the case of QoL, higher scores are better since they imply better quality of life, and for pain, lower scores are better since they imply a decrease in pain. Each subject has these scores recorded at baseline (time = 0) and then at a maximum of 5 more time points post baseline. ‘arm’ represents the treatment arm to which the subjects were assigned. The dataset is in long format.

The rest of this post is on creating a waterfall plot in R for the QoL response variable.

**Creating a waterfall plot using the barplot function in base R**

The waterfall plot is basically an ‘ordered bar chart’, where each bar represents the change from baseline response measure for the corresponding subject.

As the first step, it would be helpful if we change the format of the dataset from ‘long’ to ‘wide’. We use the reshape function to do this. Also, we retain only the QoL scores, but not the pain scores:

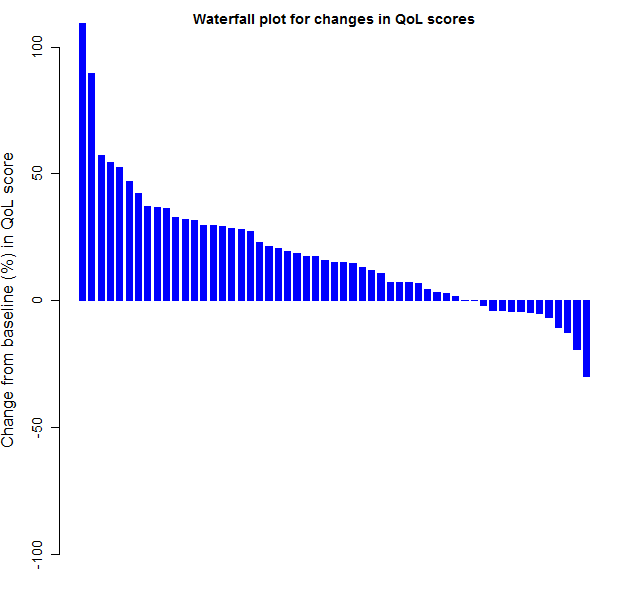
|  |  |
| --- | --- |
| 1 | qol2.wide <- reshape(dataqol2, v.names="QoL", idvar = "id", timevar = "time", direction = "wide", drop=c("date","pain")) |

For each subject, we then find the best (largest) QoL score value post baseline, compute the best percentage change from baseline and order the dataframe in the decreasing order of the best percentage changes. We also remove subjects with missing percent change values:

|  |  |
| --- | --- |
| 1  2  3  4  5 | qol2.wide$bestQoL <- apply(qol2.wide[,5:9], 1 ,**function**(x) ifelse(sum(!is.na(x)) == 0, **NA**, max(x,na.rm=**TRUE**)))  qol2.wide$bestQoL.PerChb <- ((qol2.wide$bestQoL-qol2.wide$QoL.0)/qol2.wide$QoL.0)\*100    o <- order(qol2.wide$bestQoL.PerChb,decreasing=**TRUE**,na.last=**NA**)  qol2.wide <- qol2.wide[o,] |

Create the waterfall plot… Finally!

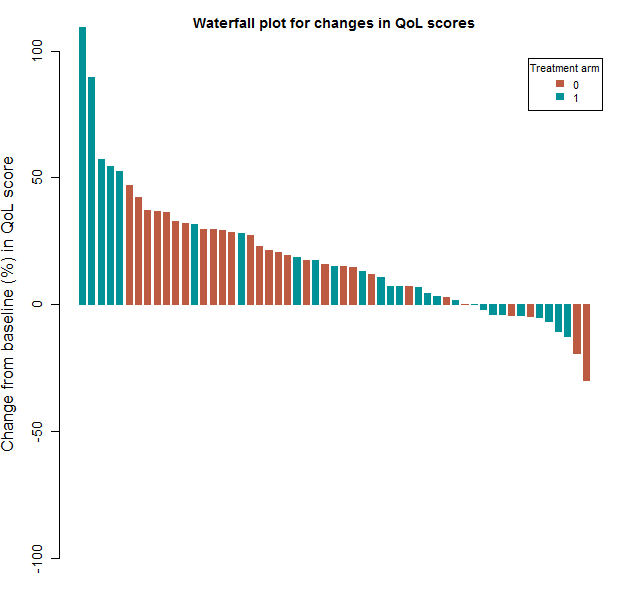
|  |  |
| --- | --- |
| 1  2  3 | barplot(qol2.wide$bestQoL.PerChb, col="blue", border="blue", space=0.5, ylim=c(-100,100),  main = "Waterfall plot for changes in QoL scores", ylab="Change from baseline (%) in QoL score",  cex.axis=1.2, cex.lab=1.4) |



Since we are depicting changes in quality of life scores, the higher the bar is in the positive direction, the better the improvement in the quality of life. So, the above figure shows that, for most subjects, there was improvement in the quality of life post baseline.

We can also color the bars differently by treatment arm, and include a legend. I used the choose\_palette() function from the excellent colorspace R package to get some nice colors.

|  |  |
| --- | --- |
| 1  2  3  4  5 | col <- ifelse(qol2.wide$arm == 0, "#BC5A42", "#009296")  barplot(qol2.wide$bestQoL.PerChb, col=col, border=col, space=0.5, ylim=c(-100,100),  main = "Waterfall plot for changes in QoL scores", ylab="Change from baseline (%) in QoL score",  cex.axis=1.2, cex.lab=1.4, legend.text=c(0,1),  args.legend=list(title="Treatment arm", fill=c("#BC5A42","#009296"), border=**NA**, cex=0.9)) |



Treatment arm 1 is associated with the largest post baseline increases in the quality of life score. Since waterfall plots are basically bar charts, they can be colored by other relevant subject attributes as well.

The above is a solution to creating waterfall plots using base R graphics function barplot. It is my aim to simultaneously also develop a solution using the ggplot2 package (and in the process, develop expertise in ggplot2). So here it is…

**Creating a waterfall plot using ggplot2**

We use the previously created qol2.wide dataframe, but in ggplot2, we also need an x variable. So:

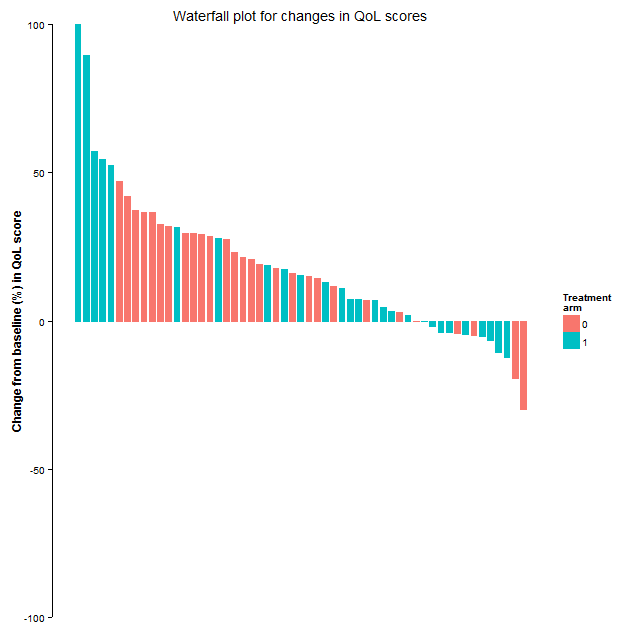
|  |  |
| --- | --- |
| 1  2 | require(ggplot2)  x <- 1:nrow(qol2.wide) |

Next we specify some plot settings, we color bars differently by treatment arm and allow the default colors of ggplot2, since I think they are quite nice. We also want to remove the x-axis, and put sensible limits for the y-axis:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | b <- ggplot(qol2.wide, aes(x=x, y=bestQoL.PerChb, fill=arm, color=arm)) +  scale\_fill\_discrete(name="Treatment\narm") + scale\_color\_discrete(guide="none") +  labs(list(title = "Waterfall plot for changes in QoL scores", x = **NULL**, y = "Change from baseline (%) in QoL score")) +  theme\_classic() %+replace%  theme(axis.line.x = element\_blank(), axis.text.x = element\_blank(), axis.ticks.x = element\_blank(),  axis.title.y = element\_text(face="bold",angle=90)) +  coord\_cartesian(ylim = c(-100,100)) |

Finally, the actual bars are drawn using geom\_bar(), and we specify the width of the bars and the space between bars. We specify stat="identity" because we want the heights of the bars to represent actual values in the data. See ?geom\_bar

|  |  |
| --- | --- |
| 1 | b <- b + geom\_bar(stat="identity", width=0.7, position = position\_dodge(width=0.4)) |



**Update**: Readers pointed out about a ‘waterfall chart’ in finance that seems to be somewhat different than the graphic discussed in this post, and they seem to use the word ‘chart’ instead of ‘plot’. [Here](http://analyticstraining.com/2015/waterfall-charts-using-ggplot2-in-r/) is some info, that also includes some R code for the waterfall chart used in finance. [Here](https://en.wikipedia.org/wiki/Waterfall_plot) is yet another plot referred to as ‘waterfall plot’ that seems to be used to display spectra.

*Waterfall* seems to be quite a popular name for plots!

Violin Plot

Dotplots are useful for the graphical visualization of small to medium-sized datasets. These simple plots provide an overview of how the data is distributed, whilst also showing the individual observations. It is however possible to make the simple dotplots more informative by overlaying them with data summaries and/or smooth distributions.

This post is about creating such superimposed dotplots in R – we first see how to create these plots using just base R graphics, and then proceed to create them using the ggplot2 R package.

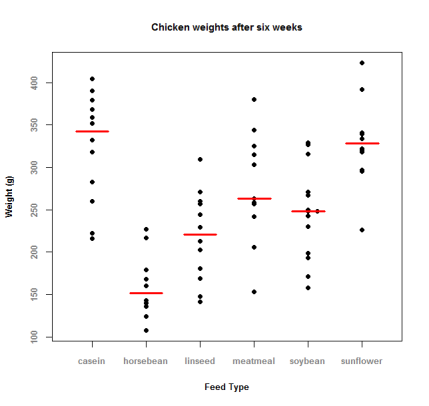
|  |  |
| --- | --- |
| 1  2  3  4  5 | ## First things first - dataset 'chickwts': Weights of  ## chickens fed with any one of six different feed types    ?chickwts  data(chickwts)  ## load the dataset |

**Graphs using base R:**

|  |  |
| --- | --- |
| 1  2  3 | ## First some plot settings    par(cex.main=0.9,cex.lab=0.8,font.lab=2,cex.axis=0.8,font.axis=2,col.axis="grey50") |

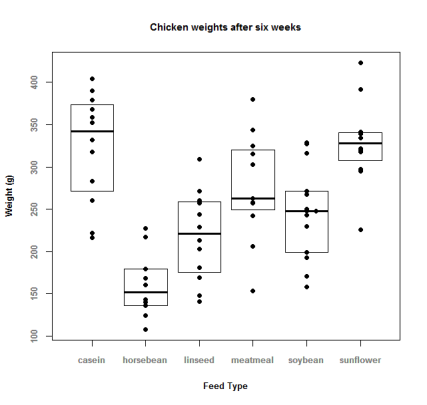
We first create a dotplot where the median of each group is also displayed as a horizontal line:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | ## Getting the dotplot first, expanding the x-axis to leave room for the line  stripchart(weight ~ feed, data = chickwts, xlim=c(0.5,6.5), vertical=**TRUE**, method = "stack", offset=0.8, pch=19,  main = "Chicken weights after six weeks", xlab = "Feed Type", ylab = "Weight (g)")    ## Then compute the group-wise medians  medians <- tapply(chickwts[,"weight"], chickwts[,"feed"], median)    ## Now add line segments corresponding to the group-wise medians  loc <- 1:length(medians)  segments(loc-0.3, medians, loc+0.3, medians, col="red", lwd=3) |



Next , we create a dotplot where the median is shown, along with the 1st and 3rd quartile, i.e., the ‘box’ of the boxplot of the data is overlaid with the dotplot:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ## Getting the dotplot first, expanding the x-axis to leave room for the box  stripchart(weight ~ feed, data = chickwts, xlim=c(0.5,6.5), vertical=**TRUE**, method="stack", offset=0.8, pch=19,  main = "Chicken weights after six weeks", xlab = "Feed Type", ylab = "Weight (g)")    ## Now draw the box, but without the whiskers!  boxplot(weight ~ feed, data = chickwts, add=**TRUE**, range=0, whisklty = 0, staplelty = 0) |

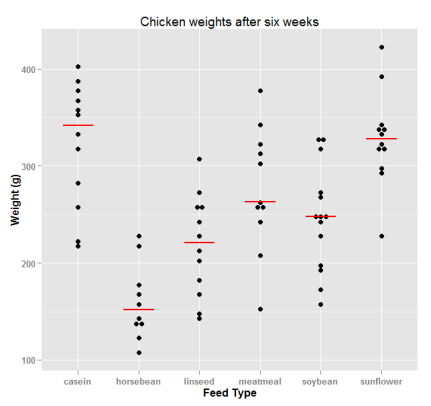
[](https://designdatadecisions.files.wordpress.com/2015/06/dotwithbox_baser.png)

**Plots similar to ones created above, but using the ggplot2 R package instead:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ## Load the ggplot2 package first  library(ggplot2)    ## Data and plot settings  p <- ggplot(chickwts, aes(x=feed, y=weight)) +  labs(list(title = "Chicken weights after six weeks", x = "Feed Type", y = "Weight (g)")) +  theme(axis.title.x = element\_text(face="bold"), axis.text.x = element\_text(face="bold")) +  theme(axis.title.y = element\_text(face="bold"), axis.text.y = element\_text(face="bold")) |

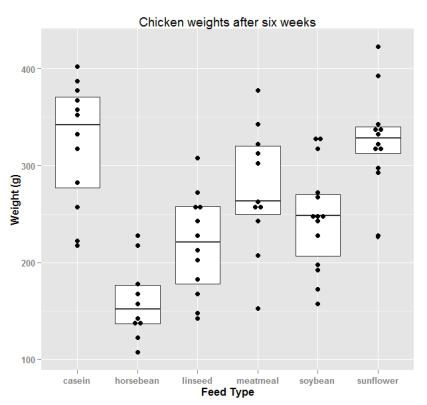
We use the stat\_summary function to plot the median line as an errorbar, but we need to define our own function that calculates the group-wise median and produces output in a format suitable for stat\_summary like so:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | ## define custom median function  plot.median <- **function**(x) {    m <- median(x)    c(y = m, ymin = m, ymax = m)  }    ## dotplot with median line  p1 <- p + geom\_dotplot(binaxis='y', stackdir='center', method="histodot", binwidth=5) +  stat\_summary(fun.data="plot.median", geom="errorbar", colour="red", width=0.5, size=1)  print(p1) |



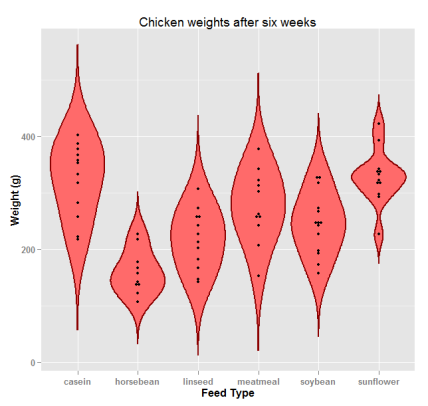
For the dotplot overlaid with the median and the 1st and 3rd quartile, the ‘box’ from the boxplot is plotted using geom\_boxplot function:

|  |  |
| --- | --- |
| 1  2  3  4 | ## dotplot with box  p2 <- p + geom\_boxplot(aes(ymin=..lower.., ymax=..upper..)) +  geom\_dotplot(binaxis='y', stackdir='center', method="histodot", binwidth=5)  print(p2) |



Additionally, let’s also plot a dotplot with a violin plot overlaid. We cannot do this in base R!

|  |  |
| --- | --- |
| 1  2  3  4  5 | ## dotplot with violin plot  ## and add some cool colors  p3 <- p + geom\_violin(scale="width", adjust=1.5, trim = **FALSE**, fill="indianred1", color="darkred", size=0.8) +  geom\_dotplot(binaxis='y', stackdir='center', method="histodot", binwidth=5)  print(p3) |



Forrest Plot

However, there is a contributed package *forestplot* that makes it very easy to make forest plots interspersed with tables – we just need to supply the right arguments to the *forestplot* function in the package. Also, one question that arose was how easy it would be to get the horizontal grey bands for alternate rows in the forest plot. This too was not very difficult. Following is a short explanation of the entire process, as well as the relevant R code:

First, we store the data for the plot, [ForestPlotData](https://designdatadecisions.files.wordpress.com/2016/07/forestplotdata.xlsx) in any convenient file format and then read it into a dataframe in R:

|  |  |
| --- | --- |
| 1  2  3 | workdir <- ""C:\\Path\\To\\Relevant\\Directory""  datafile <- file.path(workdir,"ForestPlotData.csv")  data <- read.csv(datafile, stringsAsFactors=**FALSE**) |

Then format the data a bit so that the column labels and columns match the required graphical output:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ## Labels defining subgroups are a little indented!  subgps <- c(4,5,8,9,12,13,16,17,20,21,24,25,28,29,32,33)  data$Variable[subgps] <- paste("  ",data$Variable[subgps])    ## Combine the count and percent column  np <- ifelse(!is.na(data$Count), paste(data$Count," (",data$Percent,")",sep=""), **NA**)    ## The rest of the columns in the table.  tabletext <- cbind(c("Subgroup","\n",data$Variable),                      c("No. of Patients (%)","\n",np),                      c("4-Yr Cum. Event Rate\n PCI","\n",data$PCI.Group),                      c("4-Yr Cum. Event Rate\n Medical Therapy","\n",data$Medical.Therapy.Group),                      c("P Value","\n",data$P.Value)) |

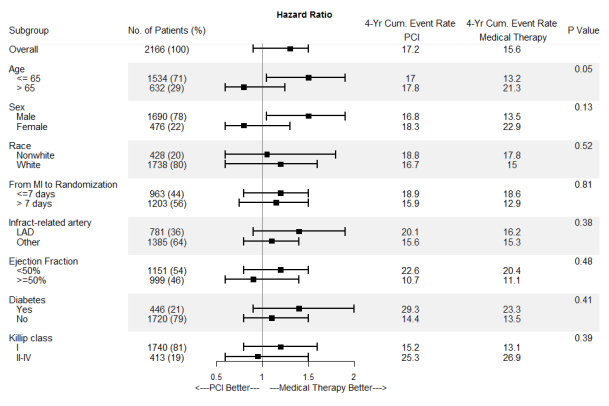
Finally, include the *forestplot* R package and call the *forestplot* function with appropriate arguments.

The way I got around to creating the horizontal band at every alternate row was by using settings for a very thick transparent line in the *hrzl\_lines* argument! See below. The *col=”#99999922″* option gives the light grey color to the line as well as sets it to be transparent.

A graphics device (here, a png file) with appropriate dimensions is first opened and the forest plot is saved to the device.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | library(forestplot)  png(file.path(workdir,"Figures\\Forestplot.png"),width=960, height=640)  forestplot(labeltext=tabletext, graph.pos=3,             mean=c(**NA**,**NA**,data$Point.Estimate),             lower=c(**NA**,**NA**,data$Low), upper=c(**NA**,**NA**,data$High),             title="Hazard Ratio",             xlab="     <---PCI Better---    ---Medical Therapy Better--->",             hrzl\_lines=list("3" = gpar(lwd=1, col="#99999922"),                            "7" = gpar(lwd=60, lineend="butt", columns=c(2:6), col="#99999922"),                            "15" = gpar(lwd=60, lineend="butt", columns=c(2:6), col="#99999922"),                            "23" = gpar(lwd=60, lineend="butt", columns=c(2:6), col="#99999922"),                            "31" = gpar(lwd=60, lineend="butt", columns=c(2:6), col="#99999922")),             txt\_gp=fpTxtGp(label=gpar(cex=1.25),                                ticks=gpar(cex=1.1),                                xlab=gpar(cex = 1.2),                                title=gpar(cex = 1.2)),             col=fpColors(box="black", lines="black", zero = "gray50"),             zero=1, cex=0.9, lineheight = "auto", boxsize=0.5, colgap=unit(6,"mm"),             lwd.ci=2, ci.vertices=**TRUE**, ci.vertices.height = 0.4)  dev.off() |

Here is the resulting forest plot:



Spider plots in oncology are used to depict changes in tumor measurements over time, relative to the baseline measurement. The resulting graph looks like the legs of a spider and hence the name. Additional information can be incorporated into the plot by varying the color and shape of points as well as the color and style of the lines.

In domains other than medical/oncology, [radar charts](https://en.wikipedia.org/wiki/Radar_chart) are sometimes also called spider plots. To clarify to readers, this post is not about the generation of radar charts.

To illustrate the generation of spider plot in R, we use as example data, the sample dataset provided along with the tumgr R package. This dataset is a sample of control arm data from a phase 3, randomized, open-label study evaluating DN-101 in combination with Docetaxel in androgen-independent prostate cancer (AIPC) (ASCENT-2). However, to illustrate the incorporation of treatment information into the plot, the subjects in this dataset were randomly placed into control and drug treatment arms. Also, the follow-up time was restricted to 240 days (8 months).

The spider plot is generated with R version 3.5.0 using package ggplot2 (version 3.0.0).

library(tumgr) ## For the example dataset

library(ggplot2)

set.seed(1234)

tumorgrowth <- sampleData

tumorgrowth <- do.call(rbind,

by(tumorgrowth, tumorgrowth$name,

function(subset) within(subset,

{ treatment <- ifelse(rbinom(1,1,0.5), "Drug","Control") Â  ## subjects are randomly placed in control or drug treatment arms

o <- order(date)

date <- date[o]

size <- size[o]

baseline <- size[1]

percentChange <- 100\*(size-baseline)/baseline

time <- ifelse(date > 240, 240, date) ## data censored at 240 days

cstatus <- factor(ifelse(date > 240, 0, 1))

})))

rownames(tumorgrowth) <- NULL

## Save plot in file

png(filename = "C:\\Path\\To\\SpiderPlot\\SpiderPlot.png", width = 640, height = 640)

## Plot settings

p <- ggplot(tumorgrowth, aes(x=time, y=percentChange, group=name)) +

theme\_bw(base\_size=14) +

theme(axis.title.x = element\_text(face="bold"), axis.text.x = element\_text(face="bold")) +

theme(axis.title.y = element\_text(face="bold"), axis.text.y = element\_text(face="bold")) +

theme(plot.title = element\_text(size=18, hjust=0.5)) +

labs(list(title = "Spider Plot", x = "Time (in days)", y = "Change from baseline (%)"))

## Now plot

p <- p + geom\_line(aes(color=treatment)) +

geom\_point(aes(shape=cstatus, color=treatment), show.legend=FALSE) +

scale\_colour\_discrete(name="Treatment", labels=c("Control", "Drug")) +

scale\_shape\_manual(name = "cstatus", values = c("0"=3, "1"=16)) +

coord\_cartesian(xlim=c(0, 240))

print(p)

dev.off()

Here is the resulting spider plot. The + symbols represent censored observations.

