Data Wrangling and Visualization

Peter Carbonetto & John Novembre

August 30, 2017

Introduction

In this tutorial, we will use the ggplot2 package to create effective visualizations of biological data. The ggplot2 package is a powerful set of plotting functions that extend the base plotting fuctions in R.

ggplot2 will also introduce you to a different way of thinking about data visualization, based on the Grammar of Graphics. *Don't be discouraged if it takes time to get used to ggplot2 and the Grammar of Graphics. This is very common!*

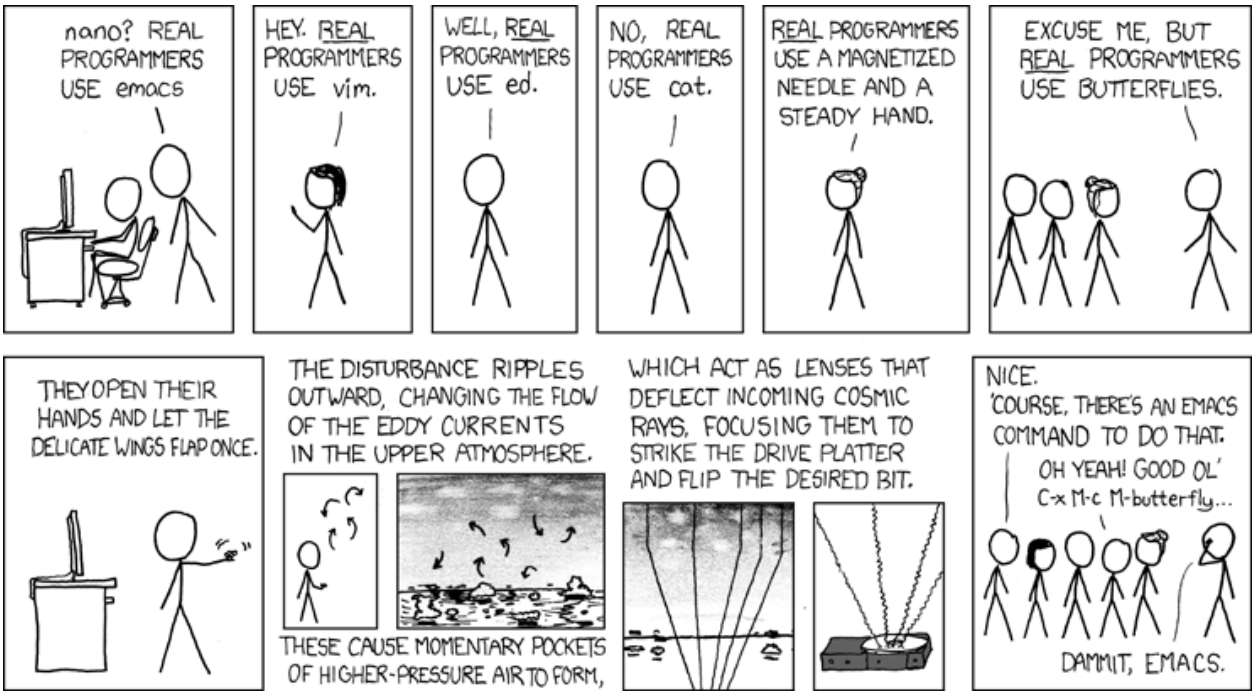
Another benefit of ggplot2 is that many help materials are available.

In this lesson, we will also see that creating effective visualizations in R *hinges on good data preparation.* In reality, good data preparation in the lab can take days or weeks, but we can still illustrate some useful data preparation practices in this workshop.

Note

This tutorial demonstrates some plotting approaches that have worked well for our research. It is important to remember, however, that there is rarely a single best approach, and you may later discover other approaches that work better for you. The R landscape is continually evolving; learning from your peers is a great way to continue to improve your R programming practice along with your research practice.

Remember, *there is no best tool—use whatever works for you.*



Background

In this tutorial, we will investigate bone-mineral density (BMD) in a population of mice known as CFW mice. (CFW is short for "Carworth Farms White", the names of the scientists who bred the first CFW mice.) In particular, we will investigate results from a genome-wide association analysis of BMD, and create plots from these results to identify genetic contributors to variation in bone-minteral density.

Setup

To follow the examples in this tutorial, you should have a reasonably up-to-date copy of R or RStudio running on your computer.

You also need to install several packages:

```
install.packages(c("ggplot2","devtools","cowplot","ggcorrplot",
                   "htmlwidgets","plotly"))
```

Once you have successfully installed the packages, load ggplot2.

```
library(ggplot2)
```

Next, we load the phenotype data from the CFW study. These data have been compiled from the Data Dryad repository for distribution as a R package. If you have not done so already, install this R package using devtools:

```
library(devtools)
install_github("pcarbo/cfwlab")
```

It may take a few minutes to install this package.

Load the package and the phenotype data:

```
library(cfwlab)
data(cfw.pheno)
```

Take a look at the data set. Each row of the `cfw.pheno` data frame is an individual, and each column represents a measurement or experimental treatment on the individual.

```
head(cfw.pheno)
```

Check the size of the data set:

```
nrow(cfw.pheno)
ncol(cfw.pheno)
```

Today, we'll start by looking at a column in this table that is relevant to understanding human skeletal diseases such as osteoporosis—bone-mineral density.

The distribution of bone-mineral density in CFW mice

Our first step is to get a visual overview of bone-mineral density in the mice we are studying. The values are stored in the column `BMD` of the dataframe. The units of BMD are mg/cm^2. (Note that this is not volumetric density—"areal" BMD is easier to measure, and is considered an accurate approximation to the true BMD.)

We can easily get a *text-based* summary with the `summary` function:

```
summary(cfw.pheno$BMD)
```

The simplest way to create a *graphical* summary is using the `quickplot` function:

```
p <- quickplot(x = cfw.pheno$BMD,
               xlab = "Bone mineral density (mg/cm^2)",
               ylab = "number of mice") +
  theme(axis.line = element_blank())
print(p)
```

When provided with a single column of numbers in the first argument, `quickplot` automatically produces a histogram. The arguments `xlab` and `ylab` specify the x and y-axis labels.

Note that ggplot2 was thoughtful enough to give us a warning about rows (samples) that were not plotted because they contained missing data.

One way ggplot2 differs from other plotting systems is that the ggplot2 functions all have a return value. This return value is a "ggplot" object:

```
class(p)
```

The plot is drawn on the screen only after invoking `print(p)`, in which `p` is the return value. (Implicitly, the `print` function is invoked whenever you type a variable name on its own, so typing `p` and `print(p)` do the same thing.)

Also, in this example we added a *layer* to the plot—a "theme layer" that alters the the overall appearance of the plot (it removes the axis lines to make a cleaner plot). In ggplot2, *all plots are built up in this way by combining plotting layers, and all layers are combined using the + operator.*

The long right-hand "tail" in the histogram seems interesting. This is an example of a "skewed distribution"—specifically it is skewed to the right, sometimes known as a positive skew. In the next section, we will create a few plots in an attempt to identify some of the genetic determinants of this high BMD.

Mapping the genetic basis of osteopetrotic bones

To map loci for excessive mineralization, we created a binary trait called "abBMD" (short for "abnormal BMD") that signals whether an individual had abnormal, or *osteopetrotic*, bones. It takes a value of 1 when BMD falls on the "long tail" of the observed distribution (BMD value greater than 90), and 0 otherwise:

```
summary(cfw.pheno$abBMD)
```

We used GEMMA to carry out a "genome-wide association study" (GWAS) for this trait; that is, we computed association p-values from the genotype data available for 79,824 SNPs on chromosomes 1–19. Let's load these data and look at the top of the data frame.

```
data(cfw.gwscan)
head(cfw.gwscan)
```

Each row is a single genetic variant in the mouse genome (a "single nucleotide polymorphism", or "SNP"), and the columns give the chromosome ("chr"), position ("pos") and a series of p-values for a test of association between variant genotype and trait value, for 8 different traits.

Our first step is to get a overview of the association results for abnormal BMD. Instead of using `quickplot` like we did before, here we model a more powerful way of creating plots using ggplot2's layering approach.

First, we prepare the data. We create a small data frame (ggplot2 always works best with data frames) with the marker number and the the BMD p-values. These will be our plotting data.

```
n    <- nrow(cfw.gwscan)
pdat <- data.frame(chr = cfw.gwscan$chr,marker = 1:n,
                   pval = cfw.gwscan$abBMD)
```

Next, let's make a simple plot with these data, and later we will see if we can improve on it.

```
p <- ggplot(data = pdat,mapping = aes(x = marker,y = pval)) +
       geom_point(color = "darkblue",shape = 20)
print(p)
```

Although this example is small, it illustrates the basic elements of a plot in ggplot2:

1. A *data frame* containing the data we want to plot.
2. An *aesthetic mapping* describing how columns of the data frame are used draw the plotting features (axes, shapes, colors, *etc.*).
3. A *geom*, short for "geometric object," that defines how the combination of data and aesthetic mapping are rendered. In short, it controls the type of plot you create.

All the plots we will create consist of different combinations of these elements.

This is not a bad first attempt—the plot shows that there are two regions in the mouse genome exhibiting very strong support for association with abnormal BMD. Also notice that in each region there are *many* strongly associated SNPs. This is a common situation, and is known as linkage disequilibrium (LD). It arises as a natural consequence of low recombination rates between markers in small populations.

One limitation of this plot is that isn't clear *where* the association signals are located. It would be nice if we could add chromosome information to this plot (notice that we do have this information in the `chr` column of the `cfw.gwscan` data frame).

One very simple way to achieve this is to vary the colour of the points according to the chromosome number:

```
p <- ggplot(data = pdat,mapping = aes(x = marker,y = pval,color = chr)) +
       geom_point(shape = 20)
print(p)
```

Suddenly we have a much more colourful plot, but it requires quite a bit of squinting to find the chromosome numbers (usually a sign that the plot could be improved).

A better way to do this—and the approach most often used for plots in GWAS papers—is to show the chromosome numbers underneath the x-axis. First, we use the `tapply` function to find the mean position of each chromosome on the x-axis:

```
x.chr <- tapply(pdat$marker,pdat$chr,mean)
print(x.chr)
```

Next, we add a new layer, `scale_x_continuous`, to customize the x-axis:

```
p <- ggplot(data = pdat,mapping = aes(x = marker,y = pval)) +
       geom_point(color = "darkblue",shape = 20) +
       scale_x_continuous(breaks = x.chr,labels = 1:19)
print(p)
```

Now we have the chromosome numbers in our plot.

This example illustrates why using a *programmatic approach* to plotting is helpful—by creating plots within the R programming environment, we have a wide array of functions at our disposal to filter, manipulate, summarize and combine the data, allowing us to generate an endless variety of plots.

However, it isn't yet clear from the plot where one chromosome ends and the other begins. This was better achieved with the rainbow-like plot above. Instead of a rainbow, let's add a column which will allow us to plot the chromosomes in alternating colors:

```
pdat <- transform(pdat,odd.chr = (as.numeric(chr) %% 2) == 1)
```

Next, we map the `odd.chr` column to the colour aesthetic, just like we did above.

```
p <- ggplot(data = pdat,mapping = aes(x = marker,y = pval,color = odd.chr)) +
  geom_point(shape = 20) +
  scale_x_continuous(breaks = x.chr,labels = 1:19)
print(p)
```

ggplot2 is smart enough to figure out that only two colours are needed because we are plotting only two different values (even and odd).

This example also demonstrates that creating sophisticated plots in ggplot requires relatively little effort *provided the data are in the right form.*

We now clearly see that the two strongest association signals are on chromosomes 5 and 11.

To mimic the colors used in a typical GWAS paper, we add a `scale_color_manual` layer that customizes the color mapping. Also, ggplot2 adds legends by default, but we clearly don't need a legend in this case, so let's remove it.

```
p <- p + scale_color_manual(values = c("skyblue","darkblue"),guide = "none")
print(p)
```

This example illustrates another important feature of ggplot2: rather than re-type the plotting code from scratch, we were able to adjust an existing plot by adding a new layer to an existing ggplot object.

Finally, let's make a few more adjustments to our plot following Best Practices. Specifically, we add more informative axis labels, add a title, remove the axis lines and unneeded axis ticks, and change the theme. Again, let's make these adjustments by adding layers to the existing ggplot object. Here we use the default theme provided by the cowplot package, which extends ggplot2 (cowplot was developed by biologist Clause Wilke). *Note:* the cowplot theme becomes the default after the package is loaded, so the `theme_cowplot` call is only needed it you would like to change the font.

```
library(cowplot)
p <- p +
  scale_y_continuous(breaks = seq(0,14,2)) +
  labs(x = "chromosome",y = "-log10 p-value",
       title = "genome-wide scan for osteopetrotic BMD") +
  theme_cowplot(font_size = 11) +
  theme(axis.line = element_blank(),axis.ticks.x = element_blank())
print(p)
```

Without too much effort, we have reproduced a publication-quality "Manhattan plot" showing the results of our genome-wide association analysis.

How to save and share your plot

There are several ways to save a figure. The best approach will depend on the aim.

For exploratory analyses, GIF and PNG are great formats because the files are easy to attach to emails or webpages and they can be viewed nearly universally. Let's save our final Manhattan plot as a PNG using the `ggsave` function:

```
ggsave("gwscan-bmd.png",plot = p,dpi = 150)
```

The best resolution (dpi = dots per inch) usually requires some trial and error, but somewhere between 100 and 200 dpi is typically sufficient for screen viewing.

Please go ahead and share this plot with one of your neighbours (e.g., by sending them an email attachment, or sharing a link to a file on Dropbox or Google Drive).

For print or publication, a GIF or PNG is usually not going to cut it; it is almost always preferrable to save the plot in a vector graphics format. Currently the most widely adopted vector graphics format is PDF:

```
ggsave("gwscan-bmd.pdf",plot = p)
```

If you open this PDF in your favourite PDF viewer (e.g., Adobe Acrobat Reader), you will see that the edges remain sharp even at very high zoom levels, in contrast to the PNG file.

Please go ahead and share this PDF with one of your neighbours.

Examining the association signal on chromosome 11

Above, our plots gave us a *genome-wide* overview of the regions of the genome contributing to variation in BMD. Next, let's try to get a finer-scale view of the association signal on chromosome 11.

First, we prepare the data. Let's create a data frame with the association results on chromosome 11 only. This can be easily done using the `subset` function:

```
pdat <- subset(cfw.gwscan,chr == 11)
```

Next, convert the positions to Megabases (Mb) by dividing by 1 million (`1e6`). This will make the base-pair positions easier to read in the plots.

```
pdat <- transform(pdat,pos = pos/1e6)
```

Now we can easily plot the p-values for chromosome 11 only, like we did it before.

```
p <- ggplot(data = pdat,mapping = aes(x = pos,y = abBMD)) +
       geom_point(color = "darkblue",shape = 20)
print(p)
```

Let's make a few improvements to this plot following Best Practices:

```
p <- ggplot(data = pdat,mapping = aes(x = pos,y = abBMD)) +
       geom_point(color = "darkblue",shape = 20) +
       labs(x = "base-pair position on chromosome 11 (Mb)",
            y = "-log10 p-value") +
       theme(axis.line = element_blank()) +
       scale_x_continuous(breaks = seq(0,120,10)) +
       scale_y_continuous(limits = c(0,15),breaks = seq(0,15,5))
print(p)
```

The strongest association signal appears to be located somewhere between 90 and 100 Mb on chromosome 11. It would be nice if we could "zoom in" on this region to get a detailed view of the association signal.

This can be done by creating an *interactive* visualization of the same data. There are several very sophisticated R packages for creating interactive visualizations. A great place to start is the plotly package because it works well with ggplot2.

The `ggplotly` function converts a ggplot object to a "plotly" object, then we use the `saveWidget` function from the `htmlwidgets` package to embed the interactive plot in an HTML file.

```
library(plotly)
library(htmlwidgets)
p.plotly <- ggplotly(p,tooltip = c("pos","abBMD"))
saveWidget(p.plotly,"plotly.html",selfcontained = TRUE)
```

Go ahead and open the webpage in your favourite browser.

A key feature of plotly is the "tooltip"; it allows you to mouse over individual data points and quickly access more details. In this example, the base-pair position and p-value are displayed in the tooltip.

Alternatively, if you are creating an R Markdown notebook, invoking the variable name in a code chunk will automatically embed the interactive plot within any webpage generated from the R Markdown notebook.

```
p.plotly
```

Examining correlation patterns within the BMD locus

In this section, we will focus on the association results between 94 Mb and 98 Mb—roughly, the region spanning the strongest association signal on chromosome 11.

In addition to getting a higher resolution view of the association signal, typically an important analysis step is to study the correlation pattern ("linkage disequilibrium") among the markers. We will explore two different ways of doing this.

First, let's zoom in on this region and plot only the markers between 94 Mb and 98 Mb.

```
pdat <- subset(pdat,pos > 94 & pos < 98)
p <- ggplot(data = pdat,mapping = aes(x = pos,y = abBMD)) +
       geom_point(color = "darkblue",shape = 20,size = 2)
print(p)
```

Now we have a higher resolution view of the association signal within this region.

Next, calculating correlations between the markers is going to involve some more intensive programming because we will compare data in a data frame (`pdat`) against a large matrix (`cfw.geno`). So bear with us for a moment.

In this next chunk, we load the genotype data, when we compute the correlations between the top SNP (the SNP with the smallest p-value) and all the other SNPs in the region. Note that squared correlations are computed by convention in GWAS. Most of the work is done in the last line, which uses function `cor` to compute correlations between the top markers and all other markers in the 95–98 Mb region.

```
data(cfw.geno)
markers    <- pdat$id
top.marker <- markers[which.max(pdat$abBMD)]
pdat$r2    <- c(cor(cfw.geno[,top.marker],cfw.geno[,markers]))^2
```

We now have a new column in the data frame, `r2`, containing the squared correlations with the top SNP:

```
head(pdat)
summary(pdat$r2)
```

Having done the work to prepare the data in a data frame, adding the correlations to the plot is straightforward—all we need to do is adjust the aesthetic mapping.

```
p <- ggplot(data = pdat,mapping = aes(x = pos,y = abBMD,color = r2)) +
       geom_point(size = 2)
print(p)
```

This time, ggplot2 recognizes that the correlations are continuously-valued, and automatically draws the points using a colour *gradient* instead of discrete colours.

However, the default choice of colors is not great, so let's choose the colours carefully following these recommendations. Similar to before, we add a "scale layer" to adjust the colour mapping. Let's also make a few other improvements to the plot.

```
clrs <- c("midnightblue","darkviolet","darkorchid","maroon",
          "tomato","orange","gold","yellow")
p <- p + scale_color_gradientn(colors = clrs) +
       scale_x_continuous(breaks = 90:104) +
       scale_y_continuous(limits = c(0,15),breaks = seq(0,14,2)) +
       labs(x = "base-pair position on chromosome 11 (Mb)",
            y = "-log10 p-value") +
       theme(axis.line = element_blank())
print(p)
```

A careful choice of colors can make the difference between an unsatisfactory plot and a more effective one.

This sort of plot is very common in GWAS for visualizing fine-scale patterns of LD.

Here, we clearly see that almost all the strongest associations are highly correlated with the top SNP. It is also interesting that there are some less strongly correlated SNPs near 95 Mb that also have very small p-values; these observations suggest that another SNP in this region may be independently associated with abnormal BMD.

One question that is still unanswered is why the association p-values drop off very suddenly at around 95 Mb, and again around 97 Mb. To better understand what is happening in this region of the genome, let's visualize the correlations between pairs of SNPs, not just the top SNP. Since visualizing correlations between >500 SNPs is difficult, let's select a small subset of SNPs that will hopefully capture the broader patterns.

```
n       <- nrow(pdat)
rows    <- seq(1,n,length.out = 32)
markers <- pdat$id[rows]
R       <- cor(cfw.geno[,markers])
```

R is a 32 x 32 matrix containing the correlations between the 32 selected SNPs in the region.

The ggcorrplot package provides a very easy-to-use interface for plotting correlation matrices:

```
library(ggcorrplot)
ldplot <- ggcorrplot(R)
print(ldplot)
```

The plot is a bit messy so let's make a few adjustments to the plot:

```
ldplot <- ggcorrplot(R,colors = c("red","white","red")) +
  scale_x_discrete(breaks = NULL) +
  scale_y_discrete(breaks = NULL) +
  theme(axis.ticks = element_blank(),axis.line = element_blank())
print(ldplot)
```

The sign of the correlation doesn't matter for studying LD (it is arbitrary based on the genotype encoding), so we adjusted the colours so that correlations of -1 and 1 are both red.

The one problem with this plot is that it is difficult to relate to the p-values without any position information. So let's add this information to the plot in the "scale" layer for the x-axis. Because ggcorrplot is based on ggplot2, we can take the same layering to make adjustments to the plot's appearance.

```
pos <- round(pdat$pos[rows],digits = 2)
ldplot <- ggcorrplot(R,colors = c("red","white","red")) +
  scale_x_discrete(labels = pos) +
  scale_y_discrete(breaks = NULL) +
  theme(axis.text.x = element_text(size = 8,color = "black"),
        axis.ticks  = element_blank(),
        axis.line   = element_blank())
print(ldplot)
```

The striking result here is that the "block" of strong correlations aligns very closely with the p-value plot; in particular, there is a large block of strongly correlated SNPs between 95 Mb and 97 Mb. This explains why we see so many strong associations with BMD; many SNPs are strongly correlated with the variant (or variants) contributing to variation in BMD.

This "block pattern" is very common, and is due to recombination not occurring uniformly across the genome. (We will explore this further in one of the exercises.)

Identifying candidate BMD genes

In this section, we will attempt to identify promising BMD gene candidates from the genetic association results. One go-to resource for this is the UCSC Genome Browser. Here, we will instead incorporate the gene annotation data directly into our plot. In so doing, we will demonstrate how to combine multiple data sets into one plot.

First, we load the gene annotations for release 38 of the Mouse Genome Assembly. These annotations were compiled from a file downloaded from the RefSeq FTP site. The `start` and `end` columns give the base-pair positions where transcription starts and ends.

```
data(genes.m38)
head(genes.m38)
```

We are only interested in the genes overlapping the 94–98 Mb region on chromosome 11. Let's create a new data frame containing the annotations for these genes only. Also, let's adjust the units of the start and end positions to Megabases (Mb) like we did for the SNP positions.

```
pdat.genes <- subset(genes.m38,chr == 11 & start < 98e6 & end > 94e6)
pdat.genes <- transform(pdat.genes,
                        start = start/1e6,
                        end = end/1e6)
print(pdat.genes)
```

We now have a data frame containing the 105 genes that overlap the region of interest.

To create the "gene track", let's plot the gene symbols at the start positions using `geom_text`. To avoid gene names overlapping each other in a jumbled mess, let's vary the vertical position of the genes by creating a new column `vpos`, and setting the y-axis position to `vpos`:

```
n <- nrow(pdat.genes)
pdat.genes$vpos <- rep(1:12,length.out = n)
geneplot <- ggplot(pdat.genes,aes(x = start,y = vpos,label = gene.symbol)) +
  geom_text(size = 3,hjust = 0,fontface = "italic")
```

Notice that we added a "label" mapping.

The cowplot package has a useful function `plot_grid` for arranging multiple plots into a single figure.

```
print(plot_grid(p + theme(legend.position = "none"),geneplot,nrow = 2))
```

We have a nice visual of the genes in and nearby the BMD locus.

The two plots don't align perfectly, even after removing the legend in the first plot. This may work for sharing your results with your colleagues, but is likely not acceptable for a publication.

Let's try a different strategy: let's add the gene symbols to the plot we created earlier. This is possible because ggplot2 allows you to specify the data set and the aesthetic mapping within the geom function call. To make sure that the gene symbols appear below the SNP p-values, let's draw the genes below the y=0 line. There is a lot going on in this code chunk, so let's walk through it carefully.

```
combinedplot <- p + geom_text(data = pdat.genes,inherit.aes = FALSE,
                              aes(x = start,y = -vpos,label = gene.symbol),
                              hjust = 0) +
  scale_y_continuous(limits = c(-12,15))
print(combinedplot)
```

Now we have the SNP p-values and gene symbols combined into a single plot.

To make this work well, we had to fix a couple of technical issues: adjust the limits of the y axis (note that ggplot2 complains about this, but allows us to do it anyway); set `inherit.aes = FALSE` so that the aesthetic mapping used to draw the p-values isn't also used to draw the genes.

Let's go the extra mile and adjust some of the plotting settings to make this plot more clear: make the text a little smaller, draw the gene symbols in italics (which is the convention for genes), and remove the y-axis ticks below the y=0 line.

```
combinedplot <- p + geom_text(data = pdat.genes,inherit.aes = FALSE,
                              aes(x = start,y = -vpos,label = gene.symbol),
                              size = 3,hjust = 0,fontface = "italic") +
  scale_y_continuous(limits = c(-12,15),breaks = seq(0,14,2))
print(combinedplot)
```

What collagen-related gene lies in the region that might be involved in this genotype-phenotype association? (See the exercises below.)

Visualizing the relationship between genotype and phenotype

Above, we identified rs29477109 as the SNP most strongly associated with abnormal BMD. In this section, we will look closely at the relationship between BMD and the SNP genotype.

First, we prepare the data. We create a new table with two columns: one for the phenotype data, and one for the genotype data (more precisely, the genotype "dosage").

```
pdat <- data.frame(BMD = cfw.pheno$BMD,dosage = cfw.geno[,top.marker])
```

Our first plot is a simple scatterplot using the "point geom".

```
p <- ggplot(data = pdat,mapping = aes(x = dosage,y = BMD)) +
  geom_point(shape = 20,size = 2,color = "darkblue")
print(p)
```

Notice that ggplot2 gives us a warning about missing values in the data frame. These points are not plotted.

This plot suggests some trend, but it is hard to make out because many of the points overlap. Perhaps it would be helpful to calculate and plot the linear trend.

The `lm` function is the "swiss army knife" for linear regression in R. Here we'll use it to fit a linear regression of BMD given genotype dosage.

```
fit <- lm(BMD ~ dosage,pdat)
summary(fit)
```

Let's add the fitted regression line to the plot. We use the `coef` function to pull out the slope and intercept of the linear regression from the `lm` output, and the "abline geom" to draw an orange, dashed line.

```
p <- p + geom_abline(slope = coef(fit)["dosage"],
                     intercept = coef(fit)["(Intercept)"],
                     linetype = "dashed",color = "darkorange")
print(p)
```

It is now a little more apparent that larger dosage values correspond to lower BMD (on average). But the effect does not appear to be very large. Indeed, in the "lm" summary above we sw that only 7.3% of variance in BMD is explained by this SNP.

Let's take a different visualization strategy by treating the genotype dosage as a discrete (or "categorical") variable. From the `cfw.map` data frame, we see that the two alleles for this variant are T and C, so the possible genotypes are TT (dosage = 0), TC and CT (dosage = 1) and CC (dosage = 2).

```
data(cfw.map)
subset(cfw.map,id == top.marker)
```

To convert the dosages to a discrete variable, we round the dosages to the maximum-probability genotype, and convert the numbers 0, 1 and 2 to genotypes TT, TC and CC.

```
pdat <- transform(pdat,genotype = factor(round(dosage)))
levels(pdat$genotype) <- c("TT","CT","CC")
summary(pdat$genotype)
```

As expected (because it is a SNP with a high MAF), most of the the mice are heterozygous CT at this SNP.

Let's try a "box plot" first, which is the canonical plot type for comparing a continuous variable and a discrete variable. A box plot is easy to create in ggplot2 because `geom_boxplot` automatically computes the relevant statistics for you.

```
p <- ggplot(data = pdat,mapping = aes(x = genotype,y = BMD)) +
  geom_boxplot(width = 0.5,na.rm = TRUE)
print(p)
```

We prefer the "violin plot" because it is easier to interpret and shows the full density, rather than the box plot's arbitrary quantiles.

```
p <- ggplot(data = pdat,mapping = aes(x = genotype,y = BMD)) +
  geom_violin(na.rm = TRUE)
print(p)
```

Let's create a new plot combining the benefits of the violin and box plots. And let's make sure we are following Best Practices by giving informative labels and titles.

```
p <- ggplot(data = pdat,mapping = aes(x = genotype,y = BMD)) +
  geom_violin(na.rm = TRUE,fill = "skyblue",color = "skyblue") +
  geom_boxplot(width = 0.1,outlier.shape = NA,na.rm = TRUE)
p <- p + labs(y = "BMD (mg/cm^2)",title = top.marker)
print(p)
```

Finally, here is a "cleaner" alternative using the `stat_summary` function. This gives us more flexibility, but requires a slightly more advanced understanding of ggplot2 to use correctly.

```
p <- ggplot(data = pdat,mapping = aes(x = genotype,y = BMD)) +
  geom_violin(na.rm = TRUE,fill = "skyblue",color = "skyblue") +
  stat_summary(fun.y = median,fun.ymin = function (x) quantile(x,0.1),
               fun.ymax = function (x) quantile(x,0.9),
               color = "black",geom = "pointrange",na.rm = TRUE)
print(p)
```

One personal favourite is the box-percentile plot from the Hmisc package, but sadly this is not currently an option in ggplot2.

The box plot and the violin plot yield a richer picture of the relationship between phenotype and genotype. In particular, we see that the TT genotype is associated with a broader distribution of high bone-mineral densities. This relationship is more striking once we visualize the full distribution.

An important take-away message from this section is that it often requires experimentation with different plotting strategies to come up with a good visualization.

Challenges/exercises

Note: These challenges are ordered in increasing level of complexity. Do not be discouraged if you have difficulty completing all the exercises—please ask the instructors for advice if you get stuck.

Part 1: Follow-up on the BMD locus

1. Look up the BMD locus on chromosome 11 in the UCSC Genome Browser and identify candidate BMD genes based on the association signal. What collagen-related gene lies in the region that might be involved in this genotype-phenotype association?

Part 2: Explore and interpret other GWAS results

The `cfw.gwscan` data frame contains association results for abnormal BMD and 7 other traits.

2. Use the `help` function in R to read the cfwlab package documentation and identify all the mouse traits that might be useful for understanding musculoskeletal diseases in humans.
3. What p-value does a -log10(p-value) of 5 correspond to? a) 0.001 b) 0.0001 c) 0.00001 d) 0.000001
4. Out of all the traits (other than "abnormal BMD"), which trait has the strongest support for a genetic association?
5. How many genetic associations for soleus muscle weight have "strong" statistical support, specifically with log10(p-values) > 6? How many distinct regions of the genome are strongly associated with soleus muscle weight at this p-value threshold?
6. Using the plotting techniques presented above, identify a "quantitative trait locus" (QTL) with a strong association signal and identify gene(s) lying near the association signal that might explain the association.

Part 3: Devise a visualization to compare the distribution of BMD in CFW mice verus HMDP mice

7. Compare the distribution of BMD in the CFW population against BMD measured in the HMDP panel to better understand whether "osteopetrotic" BMD might be particular to CFW mice. Run `data(cfw.pheno)`and `data(hmdp.pheno)` to load the data from the cfwlab package for this exercise.

Is the "long tail" of high BMD particular to the CFW population? *Advice:* There are many ways to compare histograms. Above, we arranged multiple plots in a single figure and combined multiple data sets into a single plot. Another strategy is to combine the data sets into a single data frame, and use define the aesthetic mapping in a clever way to draw the histograms in one plot.

Part 4: Add a "recombination track" to the BMD locus plot (most difficult)

8. In the *Identifying candidate BMD genes* section, we created a "gene track" to accompany the BMD locus plot. The objective of this exercise is to add recombination rate data below the BMD locus plot. Run `data(recomb.m38)` to load the the recombination rate data from the cfwlab package. To what extent to the estimated recombination rates align with the observed correlations? In particular, do recombination "hotspots" explain the sudden drop-offs in p-values around 95 Mb and 97 Mb? See Brunschwig *et al* for some interesting background on recombination hotspots in the mouse genome.