Keng GWAS Plots

R packages

OK, so Keng has some stuff she needs plotted. Hmm, plotting, I know what we need:

```
#install.packages("ggplot2")
library(ggplot2)
library(reshape2)
library(plyr)
```

These packages from Hadley Wickham are brilliant, and frankly the only reason I still use `R`. That and Rstudo's markdown magic.

Keng's Data

OK, so we all have the software we need. Time for some data. This is NPQ quantification data from Keng across many *A. thaliana* accessions.

What we do here might seem a little black magic, but it's actually not that bad. `dir()` lists files. This is the same as `ls` on the command line, for those in the know. When given a pattern, it only lists things that match that pattern. Bascially what it means is that we end up with an R list containing each of Keng's csv files, by name.

```
csvs = dir(pattern="*.csv")
print(csvs)
```

```
## [1] "BVZ0036NPQ1st2cons.csv" "BVZ0036NPQ2nd2cons.csv"
## [3] "BVZ0036NPQ3rd2cons.csv" "BVZ0036NPQ4th2cons.csv"
```

OK, so we have a list of input files, we need to read them in. `lapply` applies a function over a list, returning the results as a list. If you want, run `lapply(1:10, function(x) {x * 2})` do convince yourselves what it does.

`read.csv` does what it says on the tin, basciallly.

The second bit of this converts the altitude to the correct format. If you have weird characters in your lists of numbers ('#N/A' in this case), `read.csv` thinks they're actually categorical variables (factors in R terminology). Either fix the files, or fix it up the hacky way. I'm me, so the hacky way it is...

Finally, we just print a summary of each CSV.

```
observations = lapply(csvs, read.csv)
observations = lapply(observations,
                      function(df) {
                        # Convert the altitude to numbers, not a factor
                        df$altitude <- as.numeric(as.character(df$altitude))
                        return(df)
                      })
```

```
## Warning: NAs introduced by coercion
```

```r
for (obs in observations) {
  print(summary(obs))
}
```

```
##      EcoID           line        condition      TimePoint
##  8279   : 544   1Col-0  : 544   coastal:4832   Min.   : 20
##  149    :  32   11ME1.32:  32   inland :4832   1st Qu.: 95
##  173    :  32   122491  :  32                  Median :210
##  178    :  32   1Ag-0   :  32                  Mean   :276
##  2057   :  32   1An-1   :  32                  3rd Qu.:458
##  2150   :  32   1Bay-0  :  32                  Max.   :710
##  (Other):8960   (Other) :8960
##       NPQ          altitude
##  Min.   :0.00   Min.   :  -2.0
##  1st Qu.:0.64   1st Qu.:  64.6
##  Median :1.98   Median : 191.4
##  Mean   :1.76   Mean   : 232.7
##  3rd Qu.:2.73   3rd Qu.: 292.0
##  Max.   :6.44   Max.   :2915.0
##                 NA's   :144
##      EcoID           line        condition      TimePoint
##  8279   : 544   1Col-0  : 544   coastal:4832   Min.   : 20
##  149    :  32   11ME1.32:  32   inland :4832   1st Qu.: 95
##  173    :  32   122491  :  32                  Median :210
##  178    :  32   1Ag-0   :  32                  Mean   :276
##  2057   :  32   1An-1   :  32                  3rd Qu.:458
##  2150   :  32   1Bay-0  :  32                  Max.   :710
##  (Other):8960   (Other) :8960
##       NPQ          altitude
##  Min.   :0.00   Min.   :  -2
##  1st Qu.:0.60   1st Qu.:  67
##  Median :1.85   Median : 194
##  Mean   :1.69   Mean   : 256
##  3rd Qu.:2.57   3rd Qu.: 303
##  Max.   :5.30   Max.   :2915
##
##      EcoID           line        condition      TimePoint
##  8279   : 544   1Col-0  : 544   coastal:4832   Min.   : 20
##  149    :  32   11ME1.32:  32   inland :4832   1st Qu.: 95
##  173    :  32   122491  :  32                  Median :210
##  178    :  32   1Ag-0   :  32                  Mean   :276
##  2057   :  32   1An-1   :  32                  3rd Qu.:458
##  2150   :  32   1Bay-0  :  32                  Max.   :710
##  (Other):8960   (Other) :8960
##       NPQ           altitude
##  Min.   :0.000   Min.   :  -2
##  1st Qu.:0.698   1st Qu.:  67
##  Median :2.105   Median : 194
##  Mean   :1.875   Mean   : 256
##  3rd Qu.:2.910   3rd Qu.: 303
##  Max.   :4.170   Max.   :2915
##
##      EcoID           line        condition      TimePoint
##  8279    : 544   1Col-0   : 544   coastal:4832   Min.   : 20
##  149     :  32   11ME1.32:  32   inland :4832   1st Qu.: 95
##  173     :  32   122491   :  32                 Median :210
##  178     :  32   1Ag-0    :  32                 Mean   :276
##  2057    :  32   1An-1    :  32                 3rd Qu.:458
##  2150    :  32   1Bay-0   :  32                 Max.   :710
##  (Other):8960   (Other) :8960
##       NPQ           altitude
##  Min.   :0.00   Min.   :  -2
##  1st Qu.:0.68   1st Qu.:  67
##  Median :2.31   Median : 194
##  Mean   :1.91   Mean   : 256
##  3rd Qu.:2.87   3rd Qu.: 303
##  Max.   :4.14   Max.   :2915
##
```

Time for a plot. Let's just use the first observation for now to make life simple. So we index into the list of observation data frames (using `[[]]`, because it's a list, don't ask me why...) to grab the first observation.

```
obs1 = observations[[1]]
lines = unique(as.character(obs1$line))
summary(obs1)
```

```
##      EcoID            line        condition      TimePoint
##  8279    : 544   1Col-0   : 544   coastal:4832   Min.   : 20
##  149     :  32   11ME1.32:  32   inland :4832   1st Qu.: 95
##  173     :  32   122491   :  32                 Median :210
##  178     :  32   1Ag-0    :  32                 Mean   :276
##  2057    :  32   1An-1    :  32                 3rd Qu.:458
##  2150    :  32   1Bay-0   :  32                 Max.   :710
##  (Other):8960   (Other) :8960
##       NPQ           altitude
##  Min.   :0.00   Min.   :  -2.0
##  1st Qu.:0.64   1st Qu.:  64.6
##  Median :1.98   Median : 191.4
##  Mean   :1.76   Mean   : 232.7
##  3rd Qu.:2.73   3rd Qu.: 292.0
##  Max.   :6.44   Max.   :2915.0
##                 NA's   :144
```

We need to summarise the raw data down to means +- sd for plotting. We use the fecking amazing and impenetrable command `ddply` for this. You give it a dataframe, a list of variables you want to simplify down to, a function to run across the summary (in this case `summarise`), and a list of transformations. What this does is it trims the data frame down to a list of unique entries across the variables provided, summarising data with the functions provided.

```
obs1.sum = ddply(obs1, .(EcoID, line, condition, TimePoint, altitude), summarise,
      meanNPQ=mean(NPQ),
      sdNPQ=sd(NPQ)
      )
summary(obs1.sum)
```

```
##      EcoID            line        condition      TimePoint
##  149     :  32   11ME1.32:  32   coastal:4576   Min.   : 20
##  173     :  32   122491  :  32   inland :4576   1st Qu.: 95
##  178     :  32   1Ag-0   :  32                  Median :210
##  2057    :  32   1An-1   :  32                  Mean   :276
##  2150    :  32   1Bay-0  :  32                  3rd Qu.:458
##  2274    :  32   1Bor-1  :  32                  Max.   :710
##  (Other):8960   (Other) :8960
##     altitude         meanNPQ         sdNPQ
##  Min.   :  -2.0   Min.   :0.00   Min.   :0
##  1st Qu.:  54.1   1st Qu.:0.62   1st Qu.:0
##  Median : 158.0   Median :1.95   Median :0
##  Mean   : 234.1   Mean   :1.75   Mean   :0
##  3rd Qu.: 308.0   3rd Qu.:2.71   3rd Qu.:0
##  Max.   :2915.0   Max.   :6.44   Max.   :0
##  NA's   :144                     NA's   :9120
```

Cool, so let's plot the whole first observation in one fell swoop.

```
plt = ggplot(obs1.sum, aes(x=TimePoint, y=meanNPQ, group=interaction(condition, line), colour=condition
  geom_line() +
  scale_color_manual(values=c("#0000FF", "#FF0000")) +
  scale_x_continuous(breaks=seq(0,720,60)) +
  scale_y_continuous(limits=c(0,6)) +
  theme_classic() +
  theme(text = element_text(size=18)) +
  ylab("NPQ") +
  xlab("Seconds") +
  ggtitle("Observation 1 (all)")
pdf(paste0("plots/obs1.pdf"), width=12, height=8) # inches, for pdf device
print(plt)
dev.off()
```

```
## pdf
##   2
```

However, it seems Keng would like per-genotype plots for each line. This thousand plots bought to you by the black magic of R for loops…

(Actually, i've used `head()` to only run this on the first 3 genotypes + Col-0, otherwise it would take all week)

```
#for (geno in lines) {
for (geno in c(head(lines, n=3), '1Col-0')) {
  plt = ggplot(obs1.sum[obs1.sum$line == geno, ], aes(x=TimePoint, y=meanNPQ, group=condition, colour=co
    geom_line() +
    scale_color_manual(values=c("#0000FF", "#FF0000")) +
    scale_x_continuous(breaks=seq(0,720,60)) +
    scale_y_continuous(limits=c(0,6)) +
    theme_classic() +
    theme(text = element_text(size=16)) +
    ylab("NPQ") +
    xlab("Seconds") +
    ggtitle(geno)
  png(paste0("plots/obs1/", geno, ".png"), width=700, height=500)
  print(plt)
  dev.off()
}
```

So, what we did there was (roughly line-wise):

- Loop through each line in the dataset
- Make a basic plot per line, with x, y and groups as we expect
- tell ggplot we want a line graph
- specify line colours manually
- Specify x-axis line breaks manually, so we can have them on minute boundaries
- Set the Y axis limits manually so they're the same for each plot
- Use the classic theme – I find it clearer than the default one
- Use bigger text
- Set y & x axis lables & a title
- Open a png file for graph output
- print the graph to the file
- Close the graph file

And we're done!

But wait, there's more

Keng has 4 timepoints. Ideally, we'd have these all in one file, but ah well. Still, we want to go over each timepoint and do what we just did. I won't write out all the code, but it would look something like:

```
for (obs in 1:length(observations)) {
  this.obs = observations[[obs]]
  lines = unique(as.character(this.obs$line))
  this.obs.sum = ddply(this.obs,
                       .(EcoID, line, condition, TimePoint, altitude), summarise,
                       meanNPQ=mean(NPQ),
                       sdNPQ=sd(NPQ)
                       )
  # plot the first plot, as we did last time
  for (line in lines) {
    # plot the per-line plots as we did last time
  }
}
```