

A sampling of bioinformatics using R

Abhijit Dasgupta, PhD

Spring 2019

Expectations

1. We will explore

- the basics of using Bioconductor packages for bioinformatics
- visualizations useful for bioinformatics
- ways to annotate our graphics

2. We will not explore

- how to do bioinformatics using R
- bioinformatic workflows
- data cleaning, modeling, analytics

This area is too broad and interests too varied to cover this with any sort of breadth or depth.

A very simple example

```
fix_names <- function(d) d %>% set_names(make.names(names(.)))
clinical <- rio::import('data/BreastCancer_Clinical.xlsx') %>% fix_names()
proteome <- rio::import('data/BreastCancer_Expression.xlsx') %>% fix_names()
final_data <- clinical %>%
  inner_join(proteome, by = c('Complete.TCGA.ID' = 'TCGA_ID')) %>%
  filter(Gender == 'FEMALE') %>%
  select(Complete.TCGA.ID, Age.at.Initial.Pathologic.Diagnosis, ER.Status, starts_with("NP"))
head(final_data)
```

```
#>   Complete.TCGA.ID Age.at.Initial.Pathologic.Diagnosis ER.Status
#> 1      TCGA-A2-A0CM                                40 Negative
#> 2      TCGA-BH-A18Q                                56 Negative
#> 3      TCGA-A7-A0CE                                57 Negative
#> 4      TCGA-D8-A142                                74 Negative
#> 5      TCGA-A0-A0J6                                61 Negative
#> 6      TCGA-A2-A0YM                                67 Negative
#>   NP_958782 NP_958785 NP_958786 NP_000436 NP_958781 NP_958780
#> 1 0.6834035 0.6944241 0.6980976 0.6870771 0.6870771 0.6980976
#> 2 0.1953407 0.2154129 0.2154129 0.2053768 0.2154129 0.2154129
#> 3 -1.1231731 -1.1231731 -1.1168605 -1.1294857 -1.1294857 -1.1200168
#> 4 0.5385958 0.5422105 0.5422105 0.5349810 0.5422105 0.5422105
#> 5 0.8311317 0.8565398 0.8565398 0.8367780 0.8650092 0.8565398
#> 6 0.6558497 0.6581426 0.6558497 0.6558497 0.6512639 0.6581426
#>   NP_958783 NP_958784 NP_112598 NP_001611
#> 1 0.6980976 0.6980976 -2.6521501 -0.9843733
#> 2 0.2154129 0.2154129 -1.0357599 -0.5172257
#> 3 -1.1231731 -1.1231731 2.2445844 -2.5750648
#> 4 0.5422105 0.5422105 -0.1482049 0.2674902
```

A very simple example

```
results <- final_data %>%
  summarise_at(vars(starts_with('NP')),
    ~wilcox.test(. ~ ER.Status)$p.value)
results
```

```
#>      NP_958782 NP_958785 NP_958786 NP_000436 NP_958781 NP_958780 NP_958783
#> 1 0.6988415 0.6910103 0.6832121 0.6910103 0.6832121 0.6910103 0.6910103
#>      NP_958784 NP_112598      NP_001611
#> 1 0.6832121 0.9957714 0.0001218627
```

. is the placeholder for what's specified inside the vars().

This isn't in the right format for me to plot

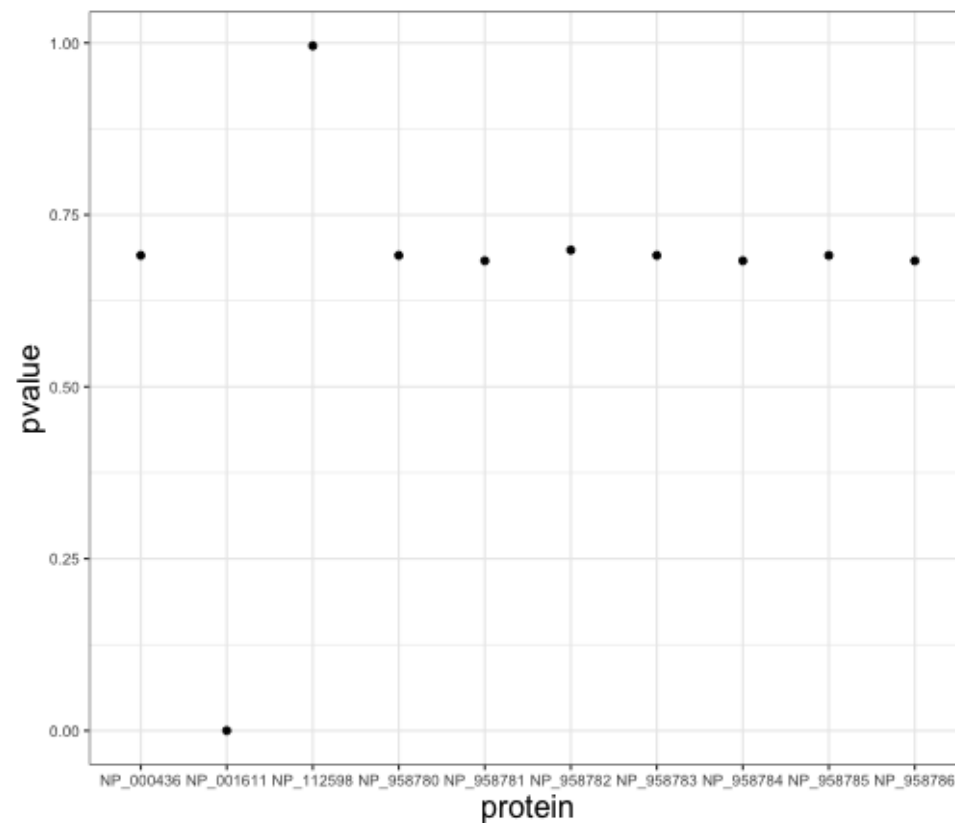
A very simple example

```
results %>% tidyr::gather(protein, pvalue)
```

```
#>      protein      pvalue  
#> 1 NP_958782 0.6988415415  
#> 2 NP_958785 0.6910103484  
#> 3 NP_958786 0.6832120796  
#> 4 NP_000436 0.6910103484  
#> 5 NP_958781 0.6832120796  
#> 6 NP_958780 0.6910103484  
#> 7 NP_958783 0.6910103484  
#> 8 NP_958784 0.6832120796  
#> 9 NP_112598 0.9957713985  
#> 10 NP_001611 0.0001218627
```

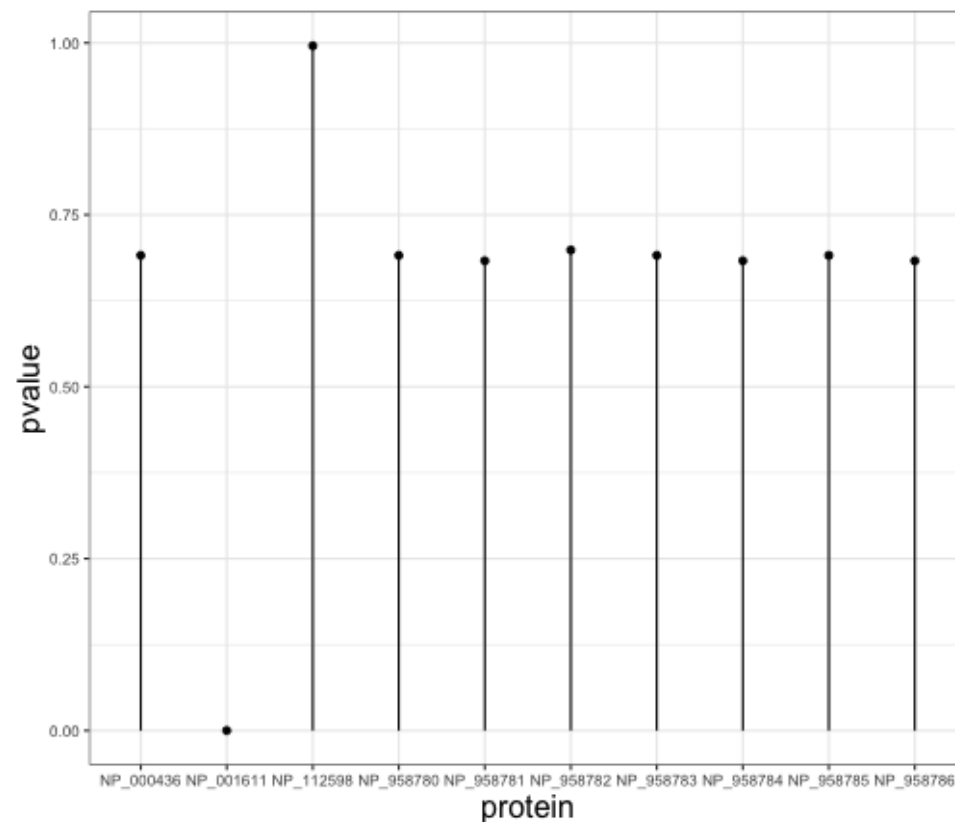
A very simple example

```
theme_439 <- theme_bw() +  
  theme(axis.title = element_text(size=16),  
        axis.text = element_text(size=8))  
  
results %>% tidyr::gather(protein, pvalue) %>%  
  ggplot(aes(x = protein, y = pvalue)) +  
    geom_point() +  
    theme_439
```



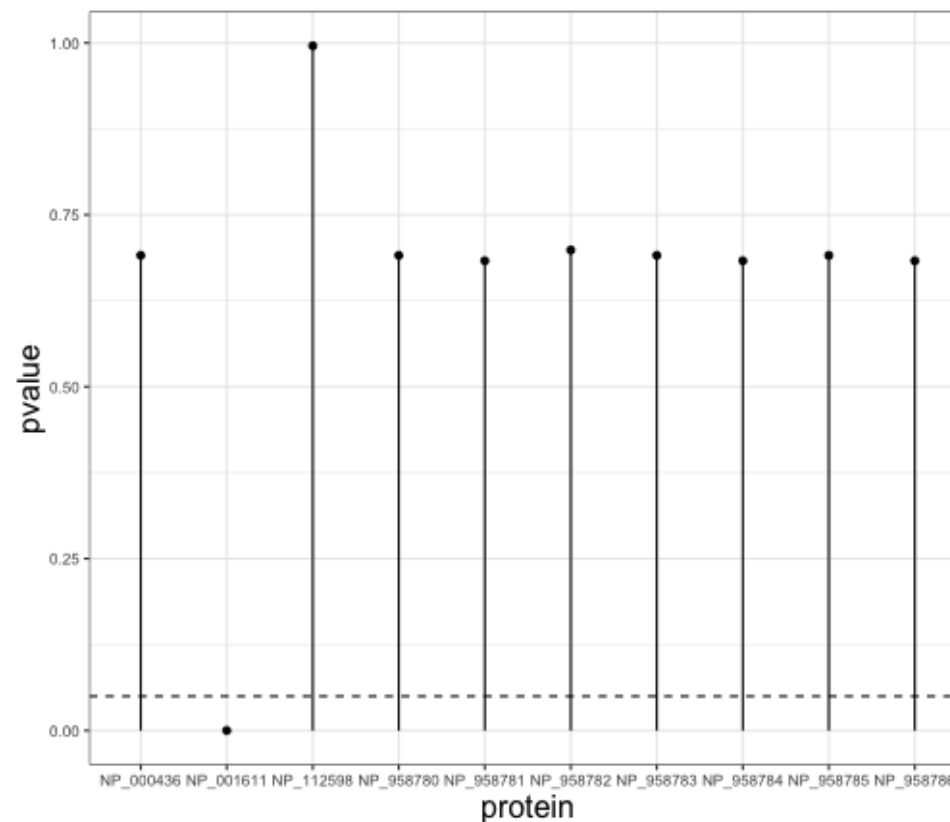
A very simple example

```
results %>% tidyr::gather(protein, pvalue) %>%  
  ggplot(aes(x = protein, y = pvalue)) +  
    geom_point() +  
    geom_segment(aes(x = protein, xend = protein,  
                     y = 0, yend = pvalue))+  
    theme_439
```



A very simple example

```
results %>% tidyr::gather(protein, pvalue) %>%  
  ggplot(aes(x = protein, y = pvalue)) +  
    geom_point() +  
    geom_segment(aes(x = protein, xend = protein,  
                     y = 0, yend = pvalue))+  
    geom_hline(yintercept = 0.05, linetype=2)+  
    theme_439
```

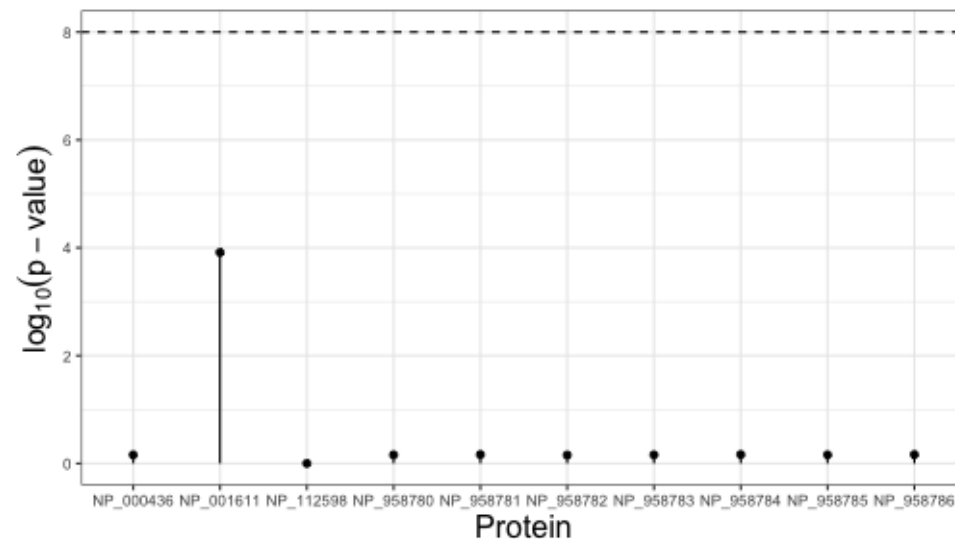


Manhattan plot

A Manhattan plot is used to visualize a set of p-values from unit-based tests

It plots the negative log p-value at each unit

```
results %>% tidyr::gather(protein, pval) %>%  
  ggplot(aes(x = protein, y = -log10(pval))) +  
  geom_point() +  
  geom_segment(aes(xend = protein, yend = 0))+  
  geom_hline(yintercept = 8, linetype=2)+  
  labs(x = 'Protein',  
        y = expression(log[10](p-value))) +  
  theme_439
```



Manhattan plot

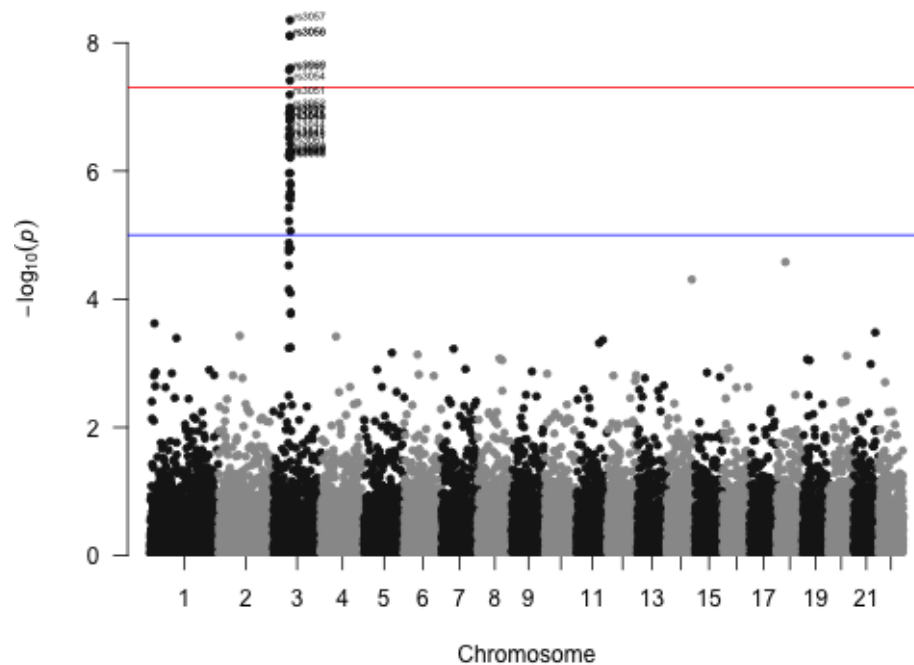
There is a specialized package for doing Manhattan plots and quantile plots for GWAS data

This package is meant to work with PLINK output, but the function is generic

```
library(qqman)  
manhattan(gwasResults)
```

Manhattan plot

```
library(qqman)
manhattan(gwasResults,
          annotatePval = 1e-6,
          annotateTop=F)
```



ggplot tours Manhattan

Data prep

```
head(gwasResults)
```

```
#>      SNP CHR BP          P  
#> 1 rs1    1  1 0.9148060  
#> 2 rs2    1  2 0.9370754  
#> 3 rs3    1  3 0.2861395  
#> 4 rs4    1  4 0.8304476  
#> 5 rs5    1  5 0.6417455  
#> 6 rs6    1  6 0.5190959
```

ggplot tours Manhattan

Data prep

```
plt_x_position <- gwasResults %>%  
  group_by(CHR) %>%  
  summarize(chr_len = max(BP)) %>%  
  mutate(tot = cumsum(chr_len) - chr_len)  
head(plt_x_position)
```

```
#> # A tibble: 6 x 3  
#>   CHR chr_len tot  
#>   <int> <dbl> <dbl>  
#> 1     1    1500     0  
#> 2     2    1191    1500  
#> 3     3    1040    2691  
#> 4     4     945    3731  
#> 5     5     877    4676  
#> 6     6     825    5553
```

ggplot tours Manhattan

Data prep

```
plt_dat <- gwasResults %>%
  left_join(plt_x_position %>% select(-chr_len),
            by = c('CHR'='CHR')) %>%
  arrange(CHR, BP) %>%
  mutate(BPcumul = BP + tot)
tail(plt_dat)
```

```
#>      SNP CHR  BP      P    tot BPcumul
#> 16465 rs16465 22 530 0.5643702 15935 16465
#> 16466 rs16466 22 531 0.1382863 15935 16466
#> 16467 rs16467 22 532 0.3936999 15935 16467
#> 16468 rs16468 22 533 0.1778749 15935 16468
#> 16469 rs16469 22 534 0.2393020 15935 16469
#> 16470 rs16470 22 535 0.2630441 15935 16470
```

ggplot tours Manhattan

Data for plotting x-axis labels

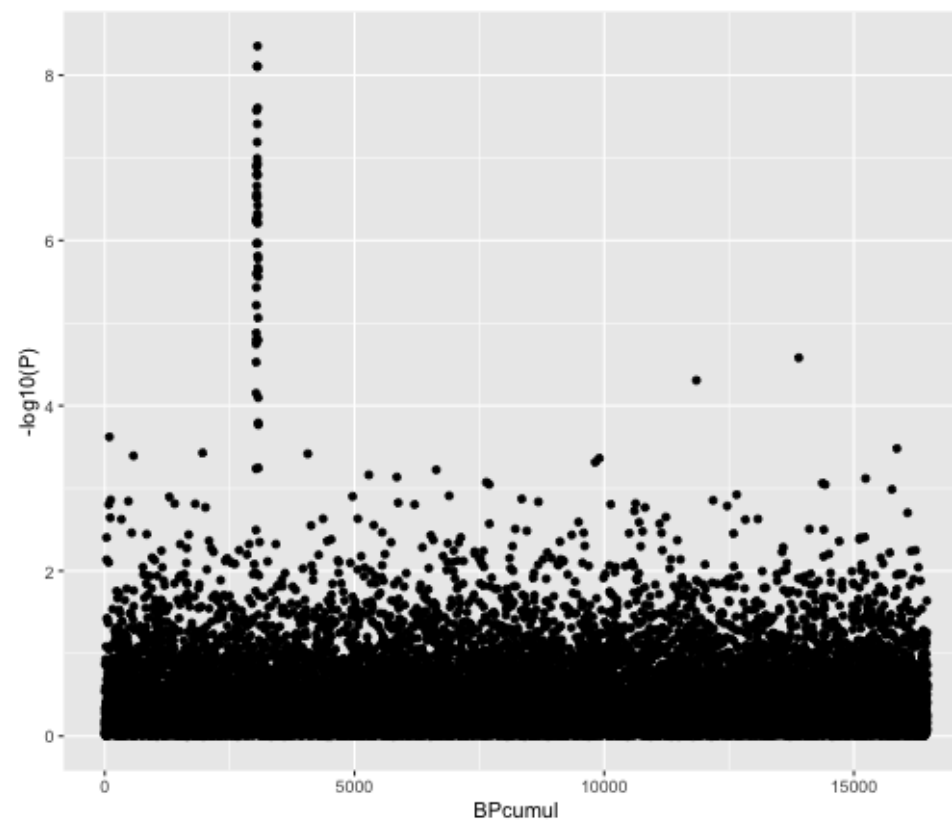
```
axisdf <- plt_dat %>%  
  group_by(CHR) %>%  
  summarize(center = (max(BPcumul)+min(BPcumul))/2)  
axisdf
```

```
#> # A tibble: 22 x 2  
#>   CHR center  
#>   <int> <dbl>  
#> 1     1  750.  
#> 2     2 2096  
#> 3     3 3212.  
#> 4     4 4204  
#> 5     5 5115  
#> 6     6 5966  
#> 7     7 6770.  
#> 8     8 7538.  
#> 9     9 8273  
#> 10    10 8982.  
#> # ... with 12 more rows
```

ggplot tours Manhattan

Plotting

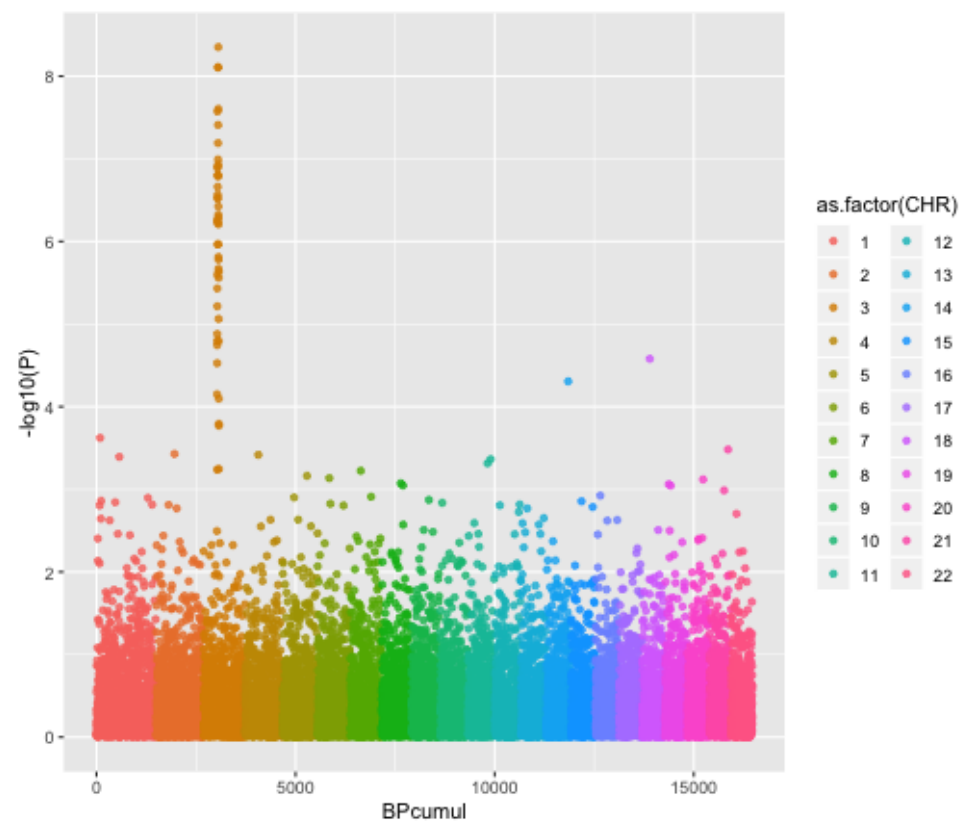
```
ggplot(plt_dat,  
      aes(x = BPcumul, y = -log10(P)))+  
  geom_point()
```



ggplot tours Manhattan

Plotting

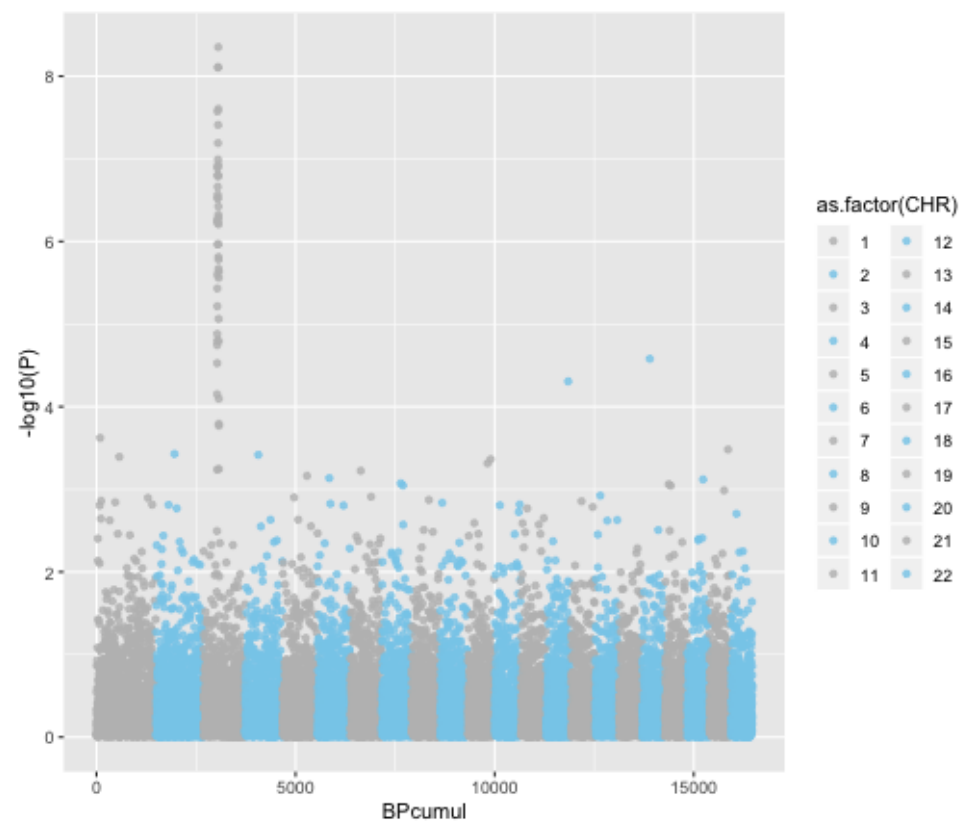
```
ggplot(plt_dat,  
      aes(x = BPcumul, y = -log10(P)))+  
  geom_point(aes(color = as.factor(CHR)),  
            alpha = 0.8, size=1.3)
```



ggplot tours Manhattan

Plotting

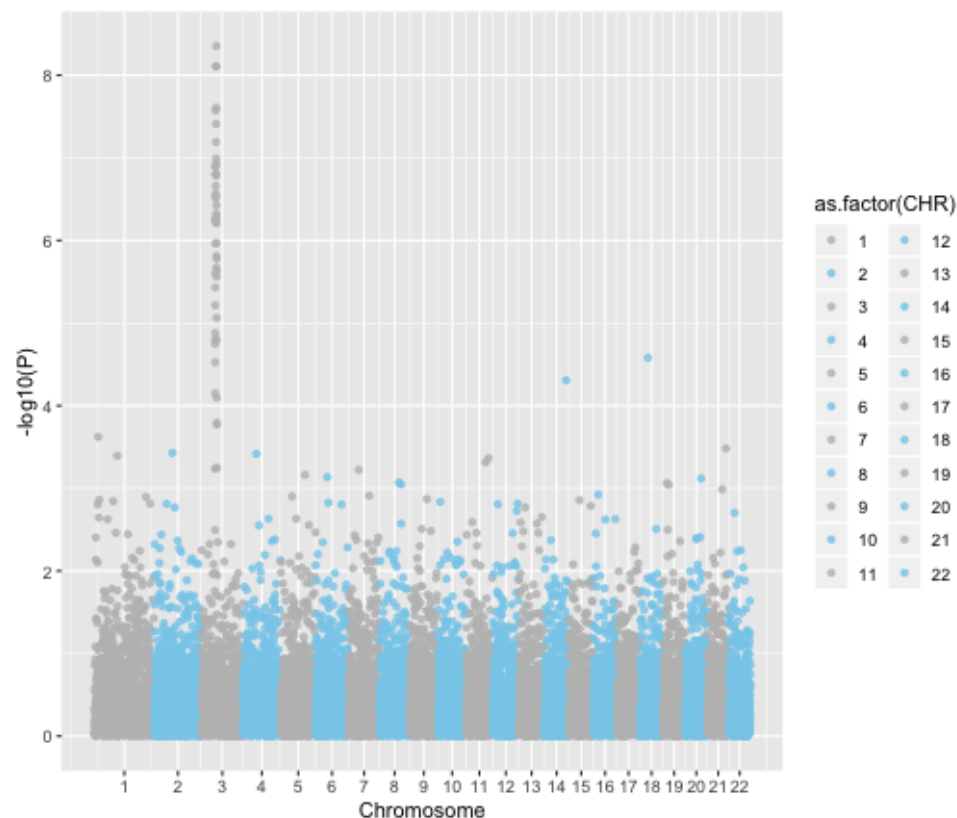
```
ggplot(plt_dat,
      aes(x = BPcumul, y = -log10(P)))+
  geom_point(aes(color = as.factor(CHR)),
            alpha = 0.8, size=1.3) +
  scale_color_manual(
    values = rep(c('grey', 'skyblue'), 22))
```



ggplot tours Manhattan

Plotting

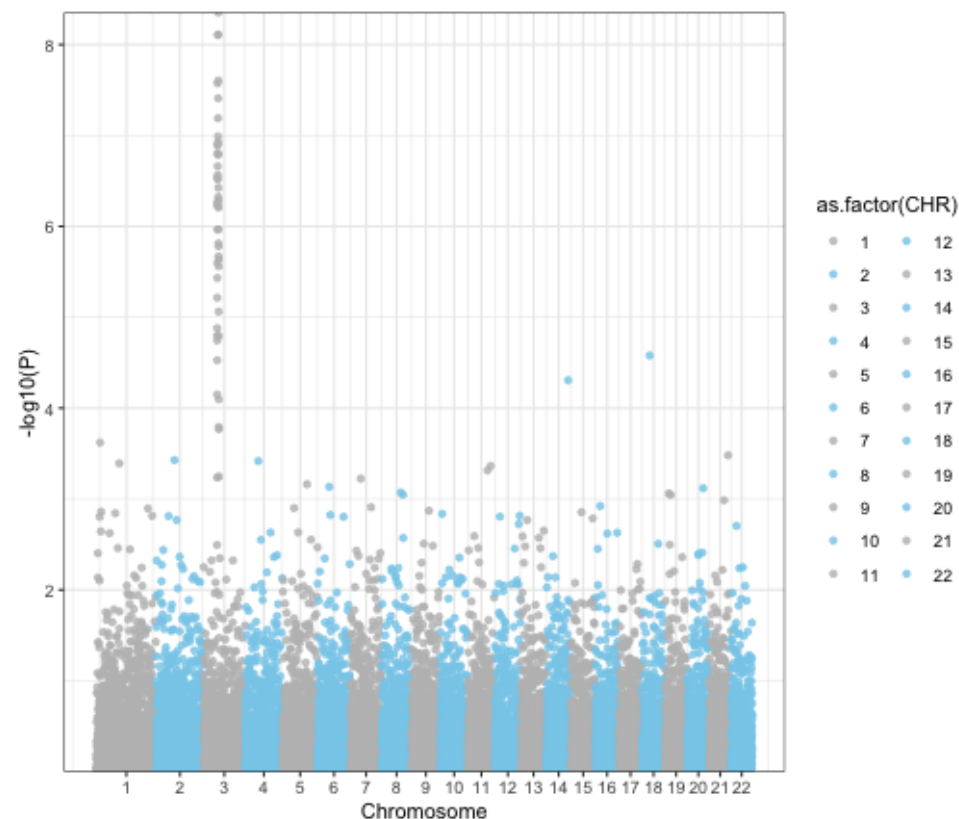
```
ggplot(plt_dat,
  aes(x = BPcumul, y = -log10(P)))+
  geom_point(aes(color = as.factor(CHR)),
    alpha = 0.8, size=1.3) +
  scale_color_manual(
    values = rep(c('grey', 'skyblue'), 22))+
  scale_x_continuous(
    name = 'Chromosome',
    breaks = axisdf$center,
    labels = axisdf$CHR
  )
```



ggplot tours Manhattan

Plotting

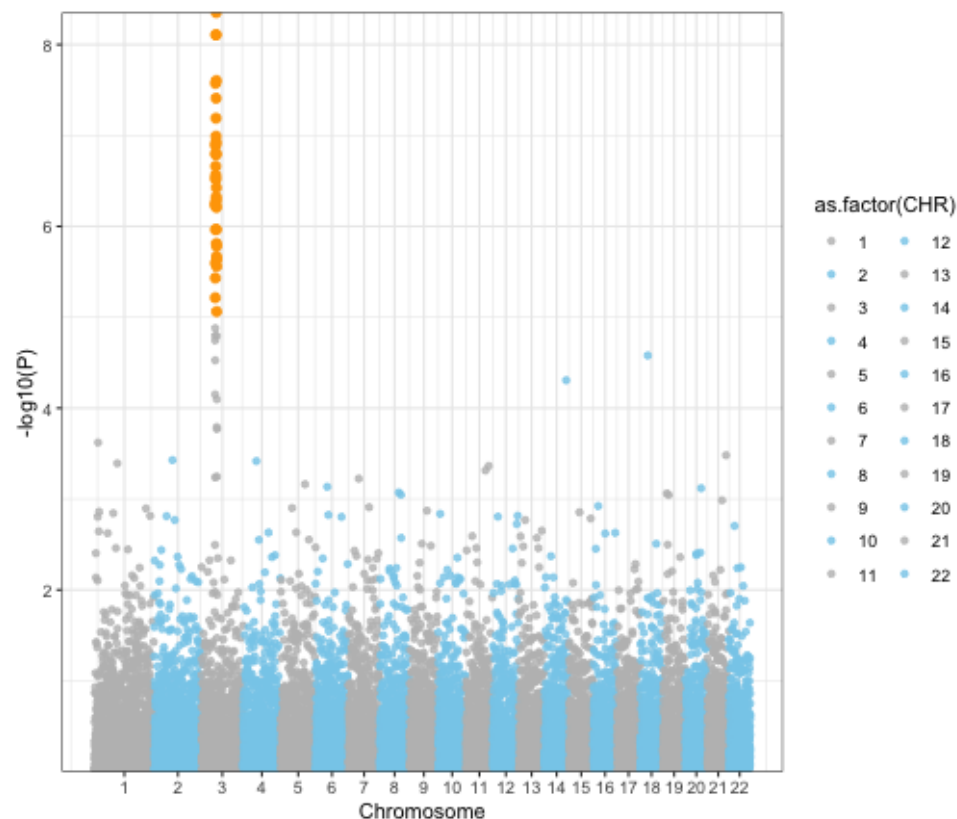
```
ggplot(plt_dat,
  aes(x = BPcumul, y = -log10(P)))+
  geom_point(aes(color = as.factor(CHR)),
    alpha = 0.8, size=1.3) +
  scale_color_manual(
    values = rep(c('grey','skyblue'), 22))+
  scale_x_continuous(
    name = 'Chromosome',
    breaks = axisdf$center,
    labels = axisdf$CHR
  ) +
  scale_y_continuous(expand = c(0,0)) +
  theme_bw()
```



ggplot tours Manhattan

Plotting

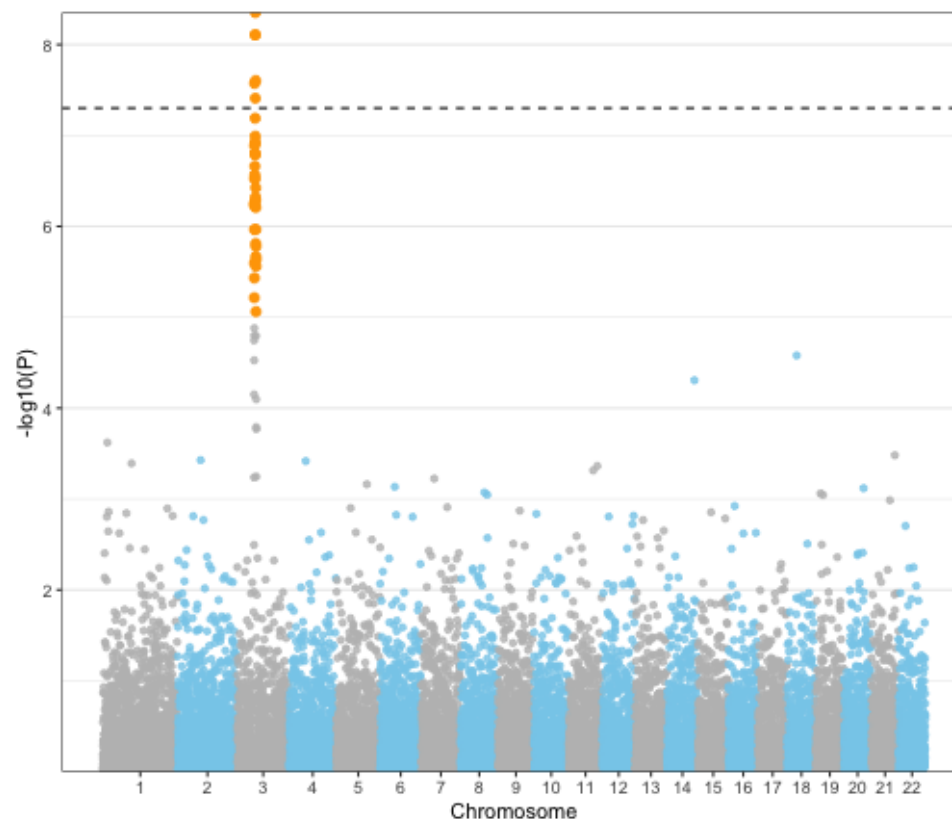
```
ggplot(plt_dat,
  aes(x = BPcumul, y = -log10(P)))+
  geom_point(aes(color = as.factor(CHR)),
    alpha = 0.8, size=1.3) +
  scale_color_manual(
    values = rep(c('grey', 'skyblue'), 22))+
  scale_x_continuous(
    name = 'Chromosome',
    breaks = axisdf$center,
    labels = axisdf$CHR
  ) +
  scale_y_continuous(expand = c(0,0)) +
  theme_bw() +
  geom_point(data = plt_dat %>% filter(P < 1e-5),
    color = 'orange',
    size=2)
```



ggplot tours Manhattan

Plotting

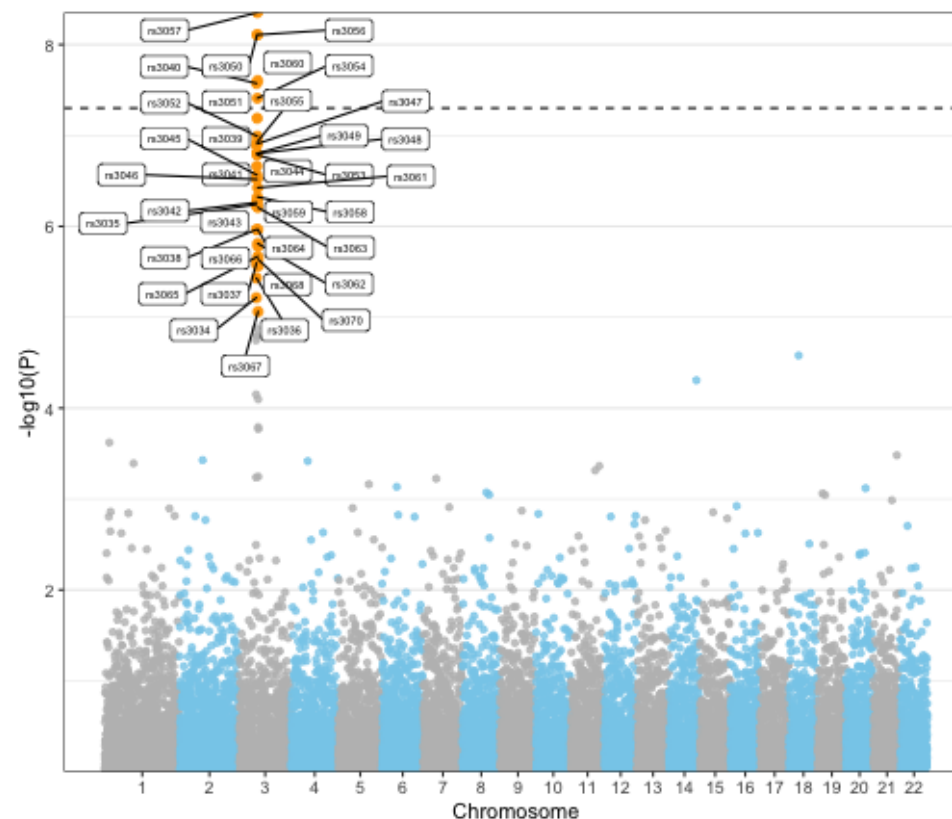
```
(plt_manhattan <- ggplot(plt_dat,
  aes(x = BPcumul, y = -log10(P)))+
  geom_point(aes(color = as.factor(CHR)),
    alpha = 0.8, size=1.3) +
  scale_color_manual(
    values = rep(c('grey','skyblue'), 22))+
  scale_x_continuous(
    name = 'Chromosome',
    breaks = axisdf$center,
    labels = axisdf$CHR
  ) +
  scale_y_continuous(expand = c(0,0)) +
  theme_bw() +
  geom_point(data = plt_dat %>% filter(P < 1e-5),
    color = 'orange',
    size=2)+
  geom_hline(yintercept = -log10(5e-8), linetype=2)
  theme(legend.position='none',
    panel.grid.major.x=element_blank(),
    panel.grid.minor.x = element_blank())
)
```



ggplot tours Manhattan

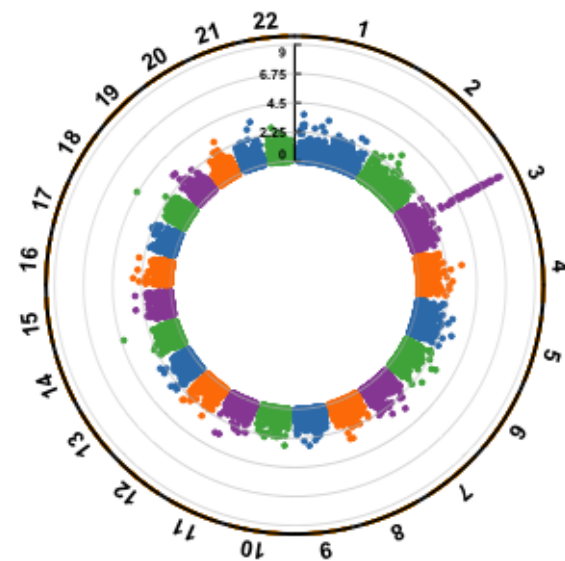
Annotation

```
library(ggrepel)
plt_manhattan +
  geom_label_repel(
    data = plt_dat %>% filter(P < 1e-5),
    aes(label = SNP),
    size = 2)
```



Circular Manhattan plots

```
library(CMplot)
CMplot(gwasResults, plot.type = 'c',
       r = 1.6,
       cir.legend = T,
       outward=T,
       cir.legend.col='black',
       cir.chr.h=.1,
       chr.den.col='orange',
       file.output=F)
```



```
#> [1] "Circular_Manhattan Plotting P..."
```


Heatmaps

Let us count the ways

There are several ways of doing heatmaps in R:

- http://sebastianraschka.com/Articles/heatmaps_in_r.html
- <https://plot.ly/r/heatmaps/>
- <http://moderndata.plot.ly/interactive-heat-maps-for-r/>
- <http://www.siliconcreek.net/r/simple-heatmap-in-r-with-ggplot2>
- <https://rud.is/b/2016/02/14/making-faceted-heatmaps-with-ggplot2/>

Some example data

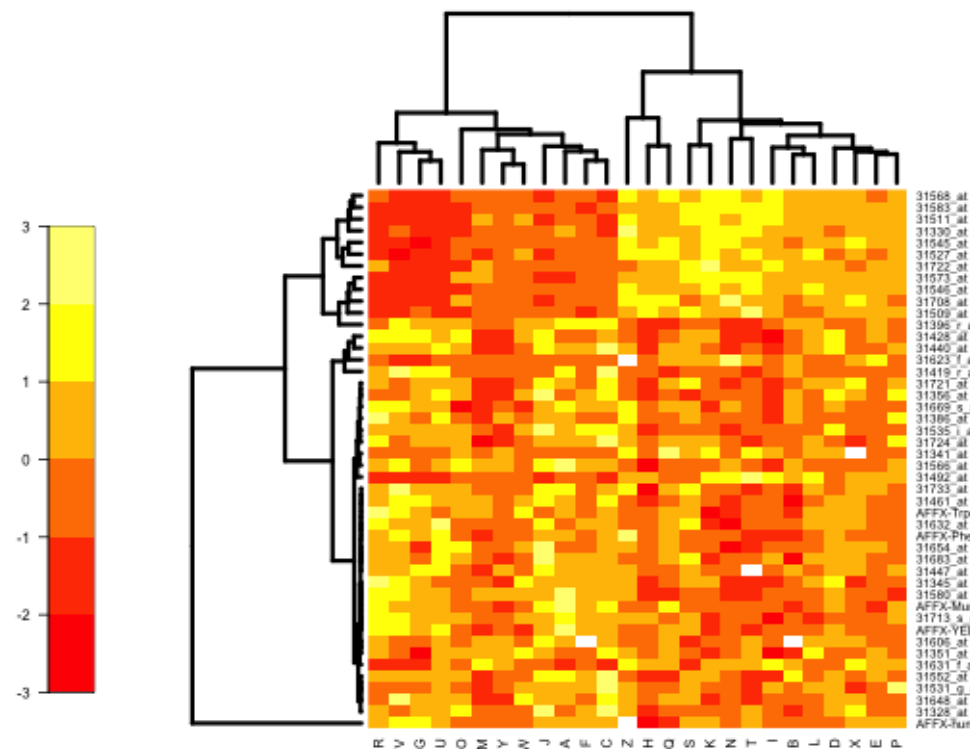
```
library(Biobase)
data(sample.ExpressionSet)
exdat <- sample.ExpressionSet
library(limma)
design1 <- model.matrix(~type, data=pData(exdat))
lm1 <- lmFit(exprs(exdat), design1)
lm1 <- eBayes(lm1) # compute linear model for each probeset
geneID <- rownames(topTable(lm1, coef=2, num=100, adjust='none', p.value=0.05))
exdat2 <- exdat[geneID,] # Keep features with p-values < 0.05

head(exdat2)
```

```
#> ExpressionSet (storageMode: lockedEnvironment)
#> assayData: 1 features, 26 samples
#> element names: exprs, se.exprs
#> protocolData: none
#> phenoData
#> sampleNames: A B ... Z (26 total)
#> varLabels: sex type score
#> varMetadata: labelDescription
#> featureData: none
#> experimentData: use 'experimentData(object)'
#> Annotation: hgu95av2
```

Using Heatplus

```
# BiocManager::install('Heatplus')
library(Heatplus)
reg1 <- regHeatmap(exprs(exdat2), legend=2, col=heat.
                    breaks=-3:3)
plot(reg1)
```



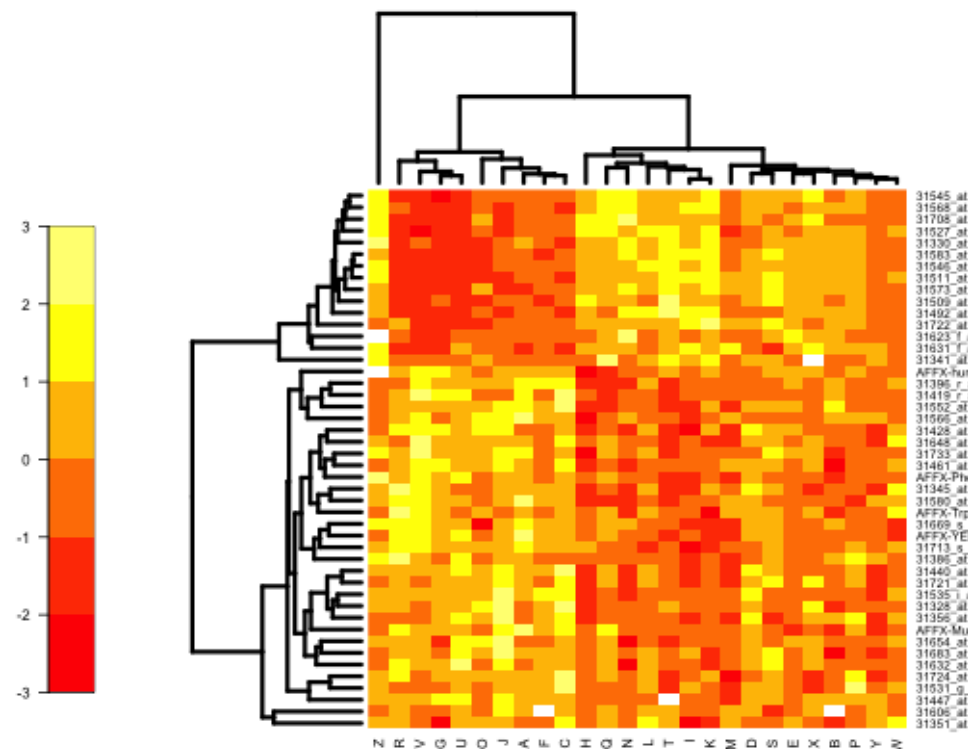
Using Heatplus

```

corrdist <- function(x) as.dist(1-cor(t(x)))
hclust.av1 <- function(x) hclust(x, method='average')
reg2 <- regHeatmap(exprs(exdat2), legend=2,
                   col=heat.colors,
                   breaks=-3:3,
                   dendrogram =
                     list(clustfun=hclust.av1,
                          distfun=corrdist))

plot(reg2)

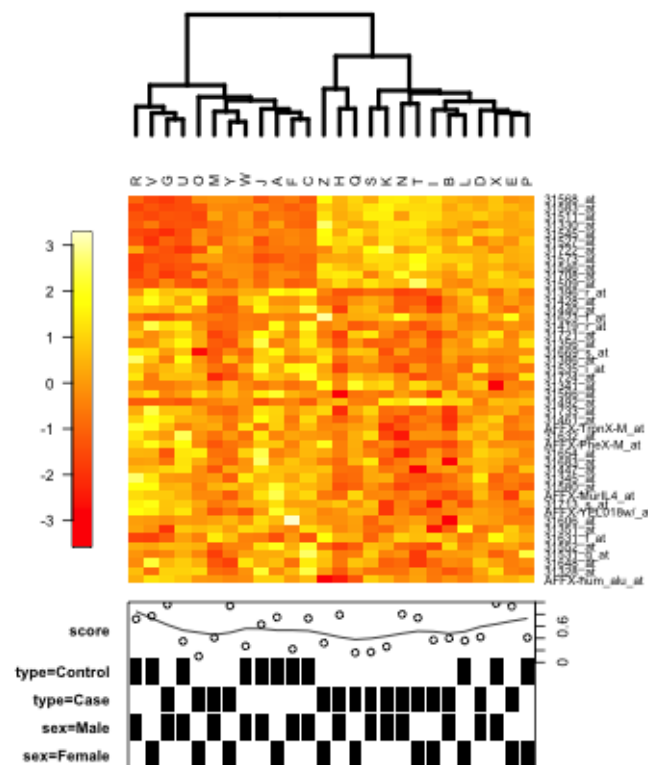
```



Using Heatplus

Adding annotations

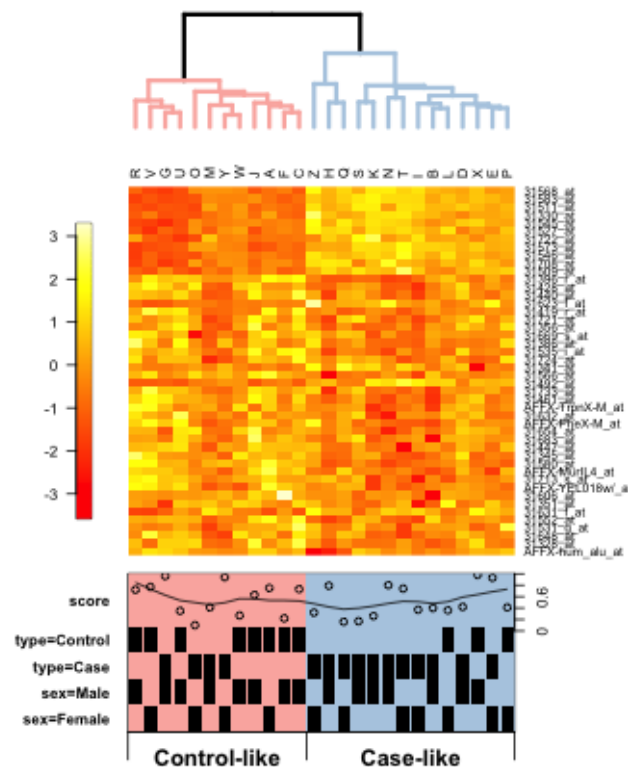
```
ann1 <- annHeatmap(exprs(exdat2),
  ann=pData(exdat2),
  col = heat.colors)
plot(ann1)
```



Using Heatplus

Adding annotations

```
ann2 <- annHeatmap(exprs(exdat2),
                   ann=pData(exdat2),
                   col = heat.colors,
                   cluster =
                     list(cuth=7500,
                          label=c('Control-like', 'Case-like')))
plot(ann2)
```



Playing with Seurat

Example data

```
library(Seurat)
# pbmc.data <- Read10X(data.dir='data/hg19/')
# pbmc <- CreateSeuratObject(counts = pbmc.data, project='pbmc3k', min.cells=3, min.features=200)
pbmc <- readRDS('data/pbmc.rds')
pbmc
```

```
#> An object of class Seurat
#> 13714 features across 2700 samples within 1 assay
#> Active assay: RNA (13714 features)
```

```
names(pbmc)
```

```
#> [1] "RNA"
```

```
slotNames(pbmc)
```

```
#> [1] "assays"      "meta.data"    "active.assay" "active.ident"
#> [5] "graphs"      "neighbors"    "reductions"   "project.name"
#> [9] "misc"        "version"      "commands"     "tools"
```

Adding QC metrics and plotting

We'll calculate mitochondrial QC metrics (percentage counts originating from mitochondrial genes)

```
pbmc[['percent.mt']] <- PercentageFeatureSet(pbmc, pattern = '^MT-')  
head(pbmc@meta.data)
```

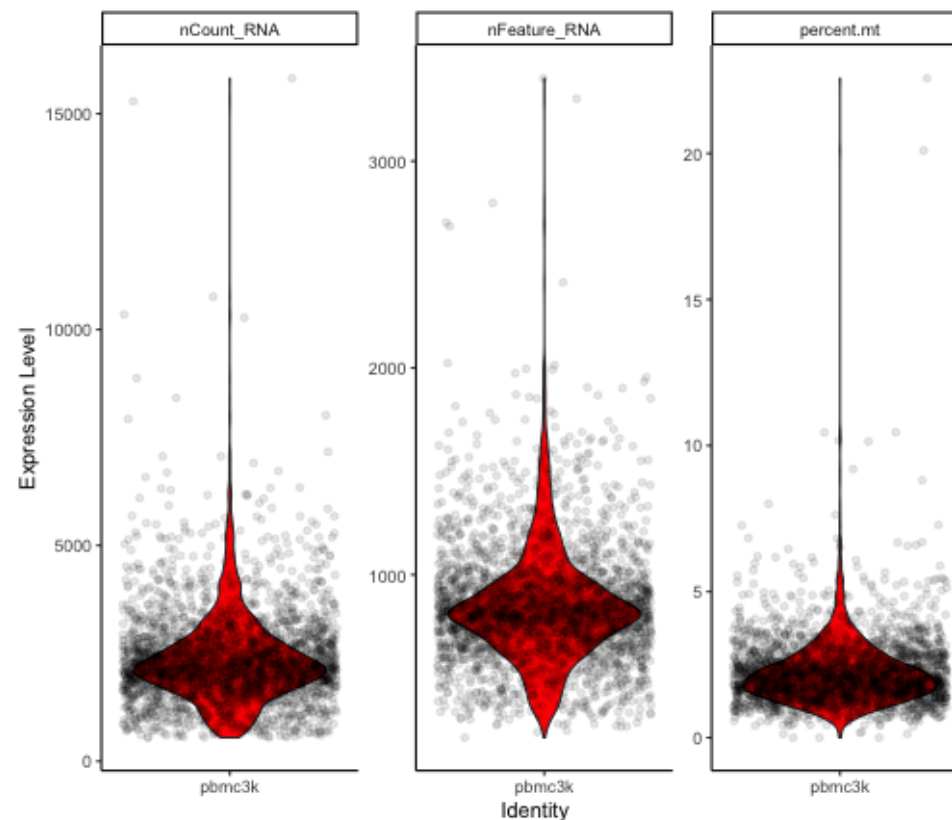
```
#>      orig.ident nCount_RNA nFeature_RNA percent.mt  
#> AAACATACAACCAC pbmc3k      2419         779  3.0177759  
#> AAACATTGAGCTAC pbmc3k      4903        1352  3.7935958  
#> AAACATTGATCAGC pbmc3k      3147        1129  0.8897363  
#> AAACCGTGCTTCCG pbmc3k      2639         960  1.7430845  
#> AAACCGTGTATGCG pbmc3k       980         521  1.2244898  
#> AAACGCACTGGTAC pbmc3k      2163         781  1.6643551
```

Visualizing metrics

```
# plt <- VlnPlot(object = pbmc,
#   features = c('nFeature_RNA',
#                 'nCount_RNA',
#                 'percent.mt'))

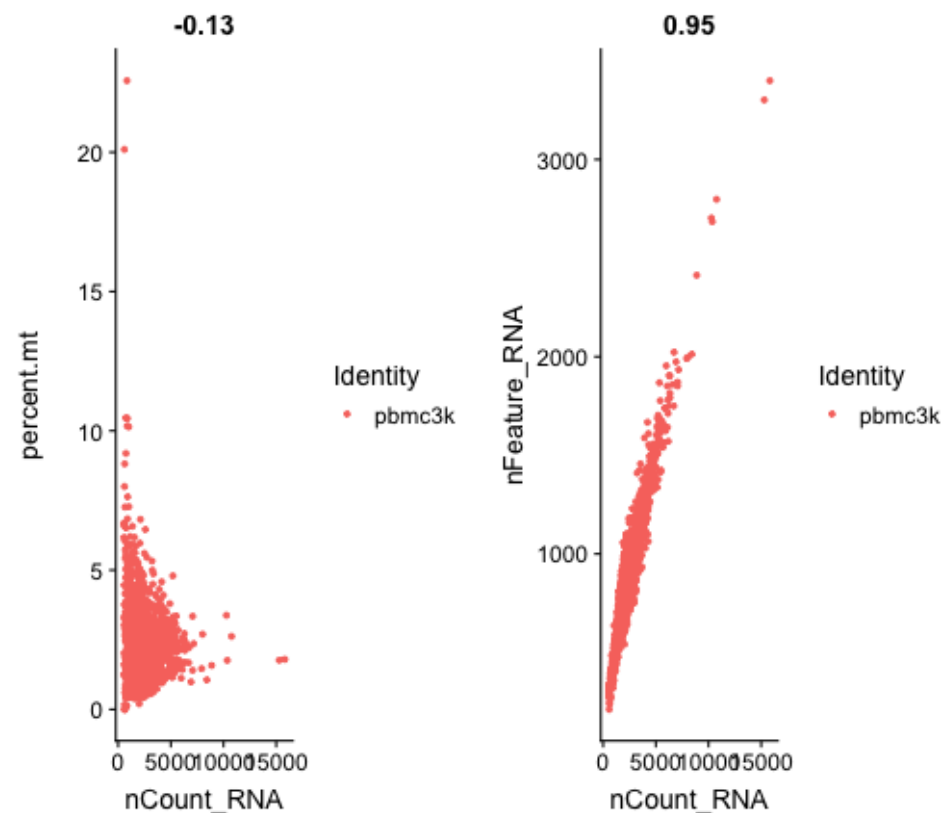
plot_data <- pbmc@meta.data %>%
  tidyr::gather(variable, value, -orig.ident)

ggplot(plot_data, aes(orig.ident, value)) +
  geom_violin(fill = 'red') +
  geom_jitter(width=0.5, alpha = 0.1) +
  facet_wrap(~variable, nrow = 1,
             scales = 'free_y') +
  labs(x = 'Identity', y = 'Expression Level') +
  theme_classic()
```



Visualizing feature-feature relationships

```
plot1 <- FeatureScatter(object = pbmc,  
                        feature1 = "nCount_RNA",  
                        feature2 = "percent.mt")  
plot2 <- FeatureScatter(object = pbmc,  
                        feature1 = "nCount_RNA",  
                        feature2 = "nFeature_RNA")  
CombinePlots(plots = list(plot1, plot2))
```



Visualizing feature-feature relationships

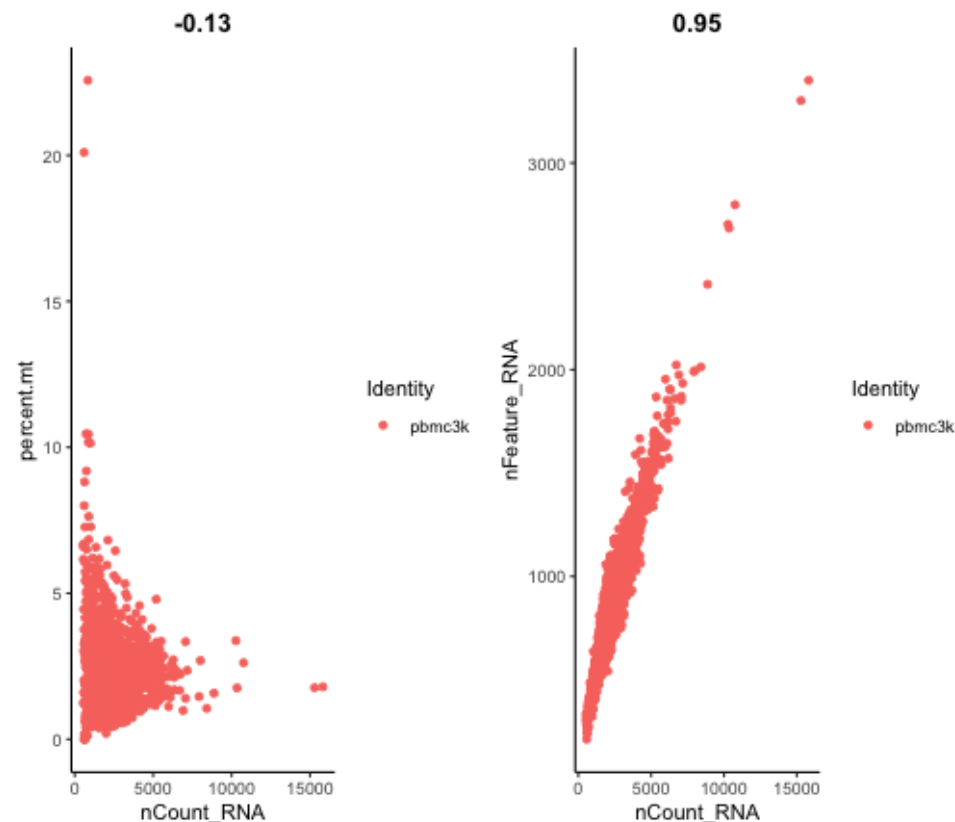
```

cormatrix <- cor(pbmc@meta.data %>% select(-orig.ident))
plt1 <-
  ggplot(pbmc@meta.data,
    aes(x = nCount_RNA,
        y = percent.mt,
        group = orig.ident,
        color = orig.ident)) +
  geom_point() +
  theme_classic() +
  labs(color = 'Identity',
        title=as.character(round(cormatrix['nCount_RNA', 'percent.mt'], 2)),
        theme(plot.title = element_text(face = 'bold', hjust = 'center')))

plt2 <-
  ggplot(pbmc@meta.data,
    aes(x = nCount_RNA,
        y = nFeature_RNA,
        group = orig.ident,
        color = orig.ident)) +
  geom_point() +
  theme_classic() +
  labs(color = 'Identity',
        title=as.character(round(cormatrix['nCount_RNA', 'nFeature_RNA'], 2)),
        theme(plot.title = element_text(face = 'bold', hjust = 'center')))

ggpubr::ggarrange(plt1, plt2, nrow = 1, ncol=2)

```



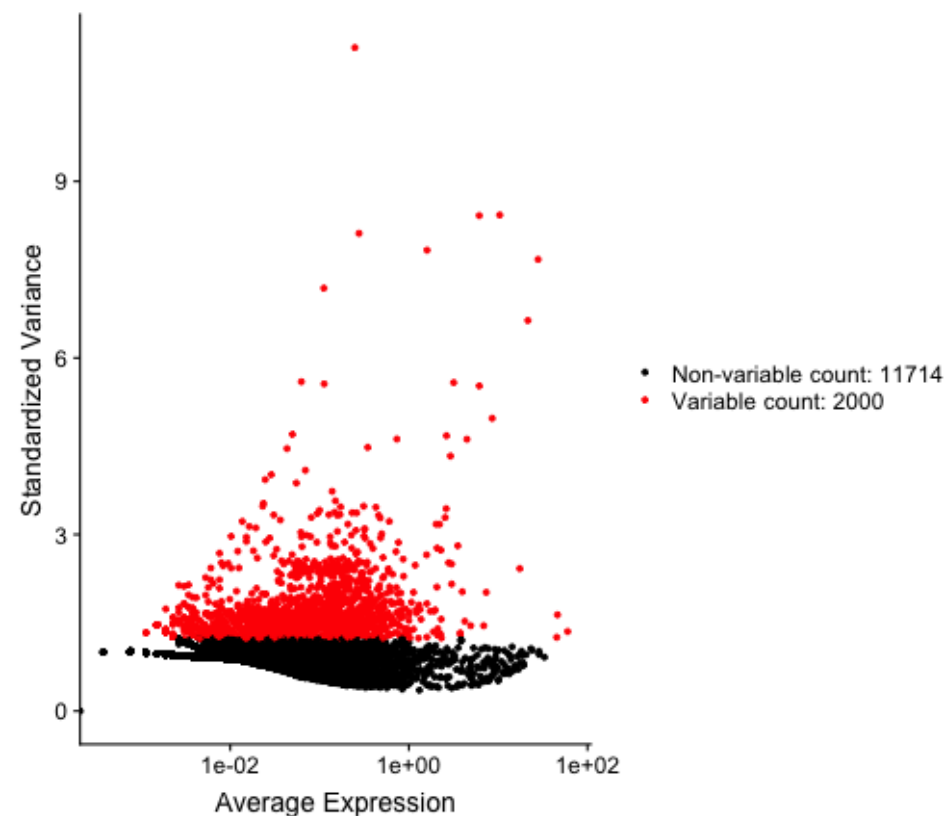
Feature selection

```
pbmc <- subset(x = pbmc,
  subset = nFeature_RNA > 200 & nFeature_RNA < 2500
pbmc <- NormalizeData(object = pbmc,
  normalization.method = "LogNorm
  scale.factor = 10000)
# This is stored in pbmc[['RNA']]@meta.features

pbmc <- FindVariableFeatures(object = pbmc,
  selection.method = "vst"
  nfeatures = 2000)

# Identify the 10 most highly variable genes
top10 <- head(x = VariableFeatures(object = pbmc), 10)

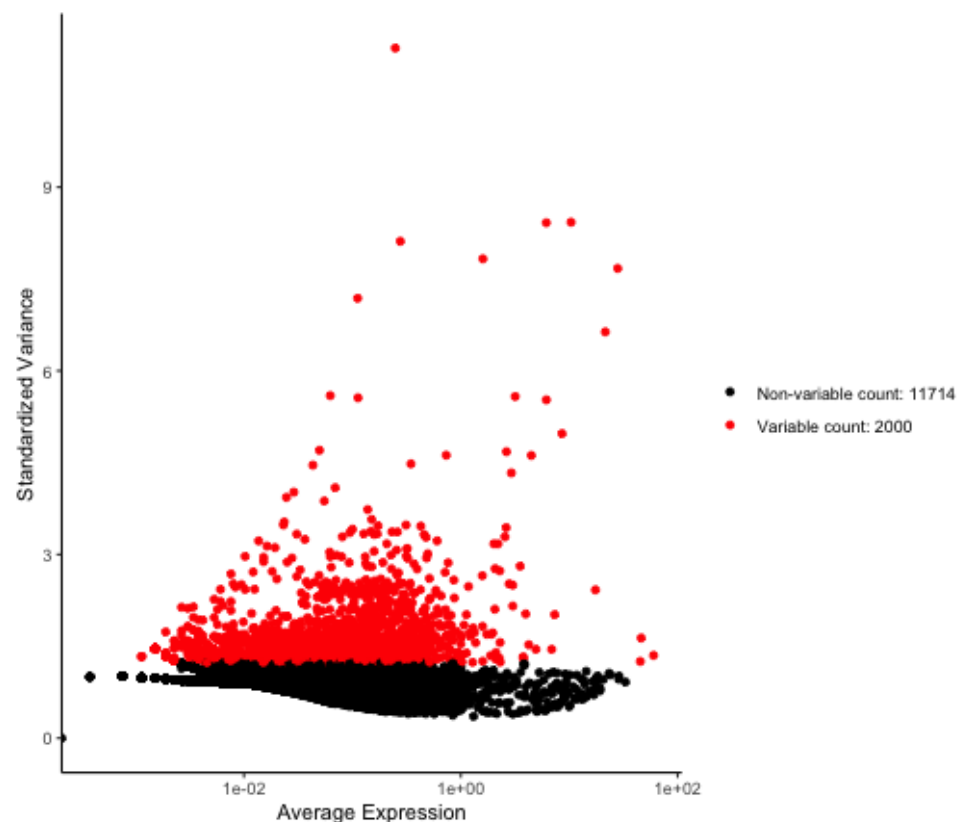
# plot variable features with and without labels
plot1 <- VariableFeaturePlot(object = pbmc)
plot1
```



Feature selection

```
plt_data <- pbmc[['RNA']]@meta.features %>%
  rownames_to_column(var='id')
topvars <- pbmc[['RNA']]@var.features
plt_data <- plt_data %>%
  mutate(indic = ifelse(id %in% topvars,
                        'Variable count',
                        'Non-variable count'))

bl <- plt_data %>%
  count(indic) %>%
  glue::glue_data("{indic}: {n}")
names(bl) <- c('Non-variable count', 'Variable count')
plt_data <- plt_data %>%
  mutate(indic = bl[indic])
plt11 <- ggplot(plt_data,
               aes(x = mean,
                   y = variance.standardized,
                   color = indic)) +
  geom_point() +
  scale_x_log10() +
  scale_color_manual(values = c('black', 'red')) +
  labs(x = 'Average Expression', y = 'Standardized Va
  theme_classic()
plt11
```



Feature selection

```
# plot2 <- LabelPoints(plot = plot1, points = top10,  
plt12 <- plt11 + ggrepel::geom_text_repel(data = plt_  
aes(label =  
color = 'bl  
  
plt12
```



There's a lot more

We'll stop our sampling here.

- Many Bioconductor packages do use ggplot, however some use base graphics
 - Faster
- Key is to find where the data is stored, and use that to create visualizations
- Bioconductor tends to create
 - One monolithic object
 - Containing different information in slots
 - combined by lists
- `slotNames` and `names` are your friends