Jennifer's R Tips and Tricks

Jennifer Liberto

20 October, 2017

Contact

For any questions/comments or if you want to play around with the data I used to try to re-create the plots and tables contact me at: jennifer.liberto@ucsf.edu
760-884-9469

Packages and Libraries

I like to use this as my `require(package)` becuase it will download, install and load them all in one quick loop. If the package already exists in my library it will skip the download and install and go straight to loading them.

```r
# look for the following packages and if they don't exist, download them from cran and install
  for (package in c('package1','package2','package3')) {
    if (!require(package, character.only=T, quietly=T, warn.conflicts = F)) {
      install.packages(package, repos = "http://cran.us.r-project.org")
      library(package, character.only=T, quietly=F, warn.conflicts = T)
    }
  }
```

Mode

R doesn't come with a mode function so this is the one I like to use

```r
# mode funciton
  getmode = function(x){
    ta = table(x)
    tam = max(ta)
    if (all(ta == tam))
         mod = NA
    else
         if(is.numeric(x))
    mod = as.numeric(names(ta)[ta == tam])
    else
         mod = names(ta)[ta == tam]
    return(mod)
  }
```

For Loops

looping through a vector

```r
# index by element
for (i in c("a","b","c")){
  somefunction(i)
}

# index by element position
for (i in 1:length(c("a","b","c"))){
  somefunction(i)
}
```

looping through a dataframe

```r
for (i in names(df)){
  somefunction(df[,i])
}
```

if loops inside for loops

```
for (i in names(df)){
  if (!(i %in% c("SubjectID", "AFR", "NAM"))) {
    somefunction(df[,i])
  } else {
    someotherfunction(df[,i])
  }
}
```

looping though a list

If you want to perform `somefunction` on variables common to more than one list, you can loop through the list and loop through the variable names.

```
dfList <- list(df1,df2)
dfNames <- c("df1","df2")
varnames <- c("var1","var2")

for(i in 1:length(dfList)){
    for (j in varnames) {
      print(paste(dfNames[i], j, sep = " "))
      somefunction(dfList[[i]][,j])
    }
}
```

advanced Loops

loop output is a vector

Say you want to remove ids from a dataframe who appear only once in the data frame. You would perform something like the following:
1. split the data frame, `df`, into a list called `dati` based on a variable, `var`
2. define an empty vector, `newvector`, that will hold the output of your loop
3. for each element in the list, if there are less than two rows place the name of that element into your vector
4. append to the vector with each iteration of the loop

```
dati <- split(df, df$var)

newvector <- c()
for (i in names(dati)){
  if (nrow(dati[[i]]) < 2) {
    newvector <- c(i,newvector)
  }
}
```

loop output is set of new variables in a data frame

Say you want to create new variables in your data frame, `df`, based on the conditions of `i` and `j`. Somehting like this will output new variables: `df$var1.var3`, `df$var1.var4`, `df$var2.var3`, `df$var2.var4` in that order.

```
vec1 <- ("var1","var2")
vec2 <- ("var3","var4")

for (i in vec1) {
  for (j in vec2) {
    newvar <- paste(i,j,sep=".")
    df[[newvar]][condiiton of i and/or j] <-"0"
    df[[newvar]][condiiton of i and/or j] <-"1"
    df[[newvar]][condiiton of i and/or j] <-"2"
    df[[newvar]][condiiton of i and/or j] <-"3"
  }
}
```

loop output is a data frame

this will output a 3X2 data frame where the two columns are ave and std.ave and the rows are the mean and sd of a, b, and c.

```
new.df <- data.frame()
for (i in c("a","b","c")){
  ave <- mean(df[,i])
  std.dev <- sd(df[,i])
  new.df <- rbind(new.df, cbind(ave,std.dev))
}
```

We will see pretty much all this in action later in the Association plots section of this tutorial.

By function

by is a wrapper for the tapply function and basically splits a data frame into parts based on an index and applies a function. for example the folliwing will give you the number of subjects that are case vs control for each ethnic group

```
by(mydata$has.asthma, mydata$group, table, useNA = 'ifany')
```

```
## mydata$group: African American
##
##   0   1
## 688 595
## ------------------------------------------------------------
## mydata$group: Mexican
##
##   0   1
## 843 537
## ------------------------------------------------------------
## mydata$group: Other Latino
##
##   0   1
## 380 272
## ------------------------------------------------------------
## mydata$group: Puerto Rican
##
##    0    1
## 1103  457
```

Export data

use `write.table` or `write.csv` to export data frames. `write.table` will output a **tab delimited** file while `write.csv` will output a **comma seperated** file. I always recommend using the options `row.names = FALSE` and `quote = FALSE`.

Grep and regex

first a lesson in regular expressions

character	legend	example	sample.match
Characters			
\d	one digit from 0 to 9	city_\d\d	city_06
\w	word character	2\w\w\w-\w	2abc-a
\s	space character	a\sb\sc	a b c
\D	not a digit	\D\D\D\D	DaDa
\W	not a word	\W\W\W\W\W	0+*)&
\S	not a whitespace	\S\S\S\S	Flip
Quantifiers			
+	one or more	\d+-\w+	12345-one
{\d}	exactly \d times	\D{4}	DaDa

{\d,\d}	\d to \d times	\w{2,5}	apple
{,\d}	\d or more times	\d{,3}	123456
More Characters			
*	zero or more times	A*B*C	AACCC
?	once or none	A?BC	BC
.	any character except a line break	a.b	abc
\.	a period	a\.b	a.b
\	escape a special character	*	*
Logic			
|	OR operand	2|3	3
(. . .)	capturing group	A(nt|pple)	Apple
\1	contents of group 1	r(e)g\1x	regex
\2	contents of group 2	(ab) + (cd) = \2 \+ \1	ab + cd = cd + ab
White Space			
\t	tab	T\tab	T ab
\n	newline	\n	
\r	carriage return	\r	
^	beginning of string or line	^abc	a
Anchors and Boundaries			
$	end of string or line	this is the end$	end
\b	word boundry	I\bbaked\pies	I baked pies
\B	not a word boundary	d.*\Bies\B.*	doggies
Character Classes			
[. . .]	one of the characters in the brackets	[A-Fa-z0-9]+	Dad03
[a-z]	range of characters between a and z	[A-D]+	CDB
[^x]	one character that is not x	[^a-z]{3}	A1!

grep from a vector or a variable in a data frame

grep uses regular expressions (regex) to capture data below will be a few examples of what grep can do

```
vec <- c("english springer spaniel","akita","shiba inu","golden retriever","lab","cocker spaniel","bermese mountain","german shorhair pointer"," german shepard","cavalier king charles spaniel","australian sheep dog","jack russel","pug")

# grab all the spaniels from the list and notice the differences in the output from each of these commands
grep("spaniel", vec)
```

```
## [1]  1  6 10
```

```
regmatches(vec, regexpr("spaniel", vec))
```

```
## [1] "spaniel" "spaniel" "spaniel"
```

```
regmatches(vec, regexpr("([a-z]+\ ?)+spaniel", vec))
```

```
## [1] "english springer spaniel"      "cocker spaniel"
## [3] "cavalier king charles spaniel"
```

```
# now grab the dogs that have no spaces in their name.  Notice the differences in the outputs from each of these commands
regmatches(vec, regexpr("[a-z]+", vec))
```

```
##  [1] "english"    "akita"      "shiba"      "golden"     "lab"
##  [6] "cocker"     "bermese"    "german"     "german"     "cavalier"
## [11] "australian" "jack"       "pug"
```

```
regmatches(vec, regexpr("^[a-z]+$", vec))
```

```
## [1] "akita" "lab"   "pug"
```

grep from a data frame

```r
# say you want to pull out all the variables that end in _before2yr into a new data frame
before2yr.df <- sapply(names(mydata), grepl, pattern = "_before2yr")

# how about finding all the variables that have data that is either 0, 1, 2, 8, 88, or NA and you want to change all the 8's and 88's to NA
for (i in names(mydata)){
            if (all(mydata[, i] %in% c(0,1,2,8,88,NA)) == TRUE){
              change.vars <- i
              mydata[, change.vars] <- gsub("^8$",NA,mydata[, change.vars])
              mydata[, change.vars] <- gsub("^88$",NA,mydata[, change.vars])
            }
          }

# how about removing all the leading and trailing whitespace in the dataframe
mydata.new <- as.data.frame(apply(mydata,2,function(x)gsub('^\\s+|\\s+$','',x)))

# how about replacing a range of values with it's average
# first you will need to find all the variables that have ranges in their data
# then you need to substiture the range for the average using the function gsubfn
# note that gsubfn does not like NA so what I did is transform NA to R4nD0m then tr
ansformed it back

for (i in names(mydata)){
            if (any(grepl("([0-9]+-[0-9]+))", mydata[, i]) == TRUE)){
              change.vars <- i
              mydata[, change.vars] <- as.character(mydata[, change.vars])
              mydata[, change.vars][is.na(mydata[, change.vars])] <- "R4nD0m"
              mydata[, change.vars] <- gsubfn("([0-9]+)-([0-9]+)",~ round((as.nume
ric(x) + as.numeric(y))/2,digits = 1),mydata[, change.vars])
              mydata[, change.vars][mydata[, change.vars] == "R4nD0m"] <- NA
            }
          }
```

Now you should have a pretty good understanding of for loops and regular expressions so lets see that in action.

Association plots

Use a for loop to generate a data frame of the logistic regressions

Lets have a look at the type of data we will be using. For this exercise, I cleaned, combineded, and subset SAGE II and GALA II.

```r
  str(mydata)
```

```
## 'data.frame':    4875 obs. of  19 variables:
##  $ SubjectID                 : Factor w/ 6396 levels "AV1001","AV1002",..: 36
37 38 43 46 49 50 51 53 56 ...
##  $ Male                      : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 1
1 1 1 1 ...
##  $ group                     : Factor w/ 4 levels "African American",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ area                      : Factor w/ 5 levels "IL","NY","PR",..: 4 4 4 4 4
4 4 4 4 4 ...
##  $ medu                      : int  1 1 1 0 1 1 1 0 0 1 ...
##  $ SESinsur                  : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ underweight               : int  0 0 NA 0 NA 0 0 0 0 0 ...
##  $ bf                        : int  0 0 NA 0 1 1 1 0 0 1 ...
##  $ preg_smoke                : int  0 0 0 1 0 0 0 0 1 0 ...
##  $ onset                     : num  13 2 4 10.5 6 10 5 2 7 10 ...
##  $ sibs_old                  : int  0 2 1 2 1 0 0 0 0 0 ...
##  $ AFR                       : num  0.93 0.836 0.815 0.84 0.858 ...
##  $ NAM                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ chest_ill_before2yr       : int  0 0 0 0 0 0 1 1 0 1 ...
##  $ cold_before2yr            : int  0 0 0 0 0 0 1 1 0 0 ...
##  $ pneumonia_before2yr       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ bronchitis_before2yr      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ bronchiolitis_RSV_before2yr: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ has.asthma                : int  1 1 1 1 1 1 1 1 1 1 ...
```

I like to define variables that will be used in my models before hand so that the code is not too bogged down with text

```
# define the elements of your logistic model
  illness = c("chest_ill_before2yr", "cold_before2yr", "pneumonia_before2yr", "bro
nchitis_before2yr", "bronchiolitis_RSV_before2yr")
  illness.abrv = c("chest_ill", "cold", "pneumonia", "bronc", "rsv")
  outcome <- "has.asthma ~"
```

Take a second to look through this for loop and see if you can follow along.

```
# subset
  df.AA <- mydata[which(mydata$group == "African American"),]
  df.PR <- mydata[which(mydata$group == "Puerto Rican"),]
  df.MX <- mydata[which(mydata$group == "Mexican"),]

# turn into a list
  dfList <- list(df.AA, df.PR, df.MX)

# for each illness in each dataframe run a logistic model and and compile the data
into a new data frame
final.df <- data.frame()
for (i in 1:length(dfList)){
  temp.df <- data.frame()
  for (j in illness) {
  if (all(dfList[[i]]$group == "African American") == TRUE) {
    form <- formula(paste(outcome,j,"+ bf + underweight + preg_smoke + sibs_old +
Male + medu + SESinsur"))
  } else {
    form <- formula(paste(outcome,j,"+ bf + underweight + preg_smoke + sibs_old +
Male + medu + SESinsur + area"))
  }
   mod <- glm(form, data = dfList[[i]], family = 'binomial')
   temp.df <- rbind(temp.df,
                    cbind(
                     OR = round(exp(coef(mod)[-1]), digits = 3),
                     LL = round(exp(confint(mod)[-1,1]), digits = 3),
                     UL = round(exp(confint(mod)[-1,2]), digits = 3),
                     N = nobs(mod),
                     p.value = summary(mod)$coef[-1, 4]
                    )[1,]
                  )
  }
  names(temp.df) <- c('OR', 'LL', 'UL', 'N', 'p.value')
  temp.df$cond <- illness.abrv
  if (all(dfList[[i]]$group == "African American") == TRUE) {
    temp.df$group <- "African American"
  } else if (all(dfList[[i]]$group == "Puerto Rican") == TRUE) {
    temp.df$group <- "Puerto Rican"
  } else {
    temp.df$group <- "Mexican"
  }
  final.df <- rbind(final.df,temp.df)
}
```

visualize the output in a kable table

in markdown kable creates some quite beautiful and customizable tables

```
# view the final data frame that we made
kable.df <- final.df[,-c(7)]
  kable(kable.df, "html", align = "c", row.names = FALSE) %>%
    kable_styling(bootstrap_options ="striped",full_width = F) %>%
    group_rows("African American", 1, 5, label_row_css = "background-color: #c1c1c1
; color: #fff;")  %>%
    group_rows("Puerto Rican", 6, 10, label_row_css = "background-color: #c1c1c1;
color: #fff;")  %>%
    group_rows("Mexican", 11, 15, label_row_css = "background-color: #c1c1c1; colo
r: #fff;")
```

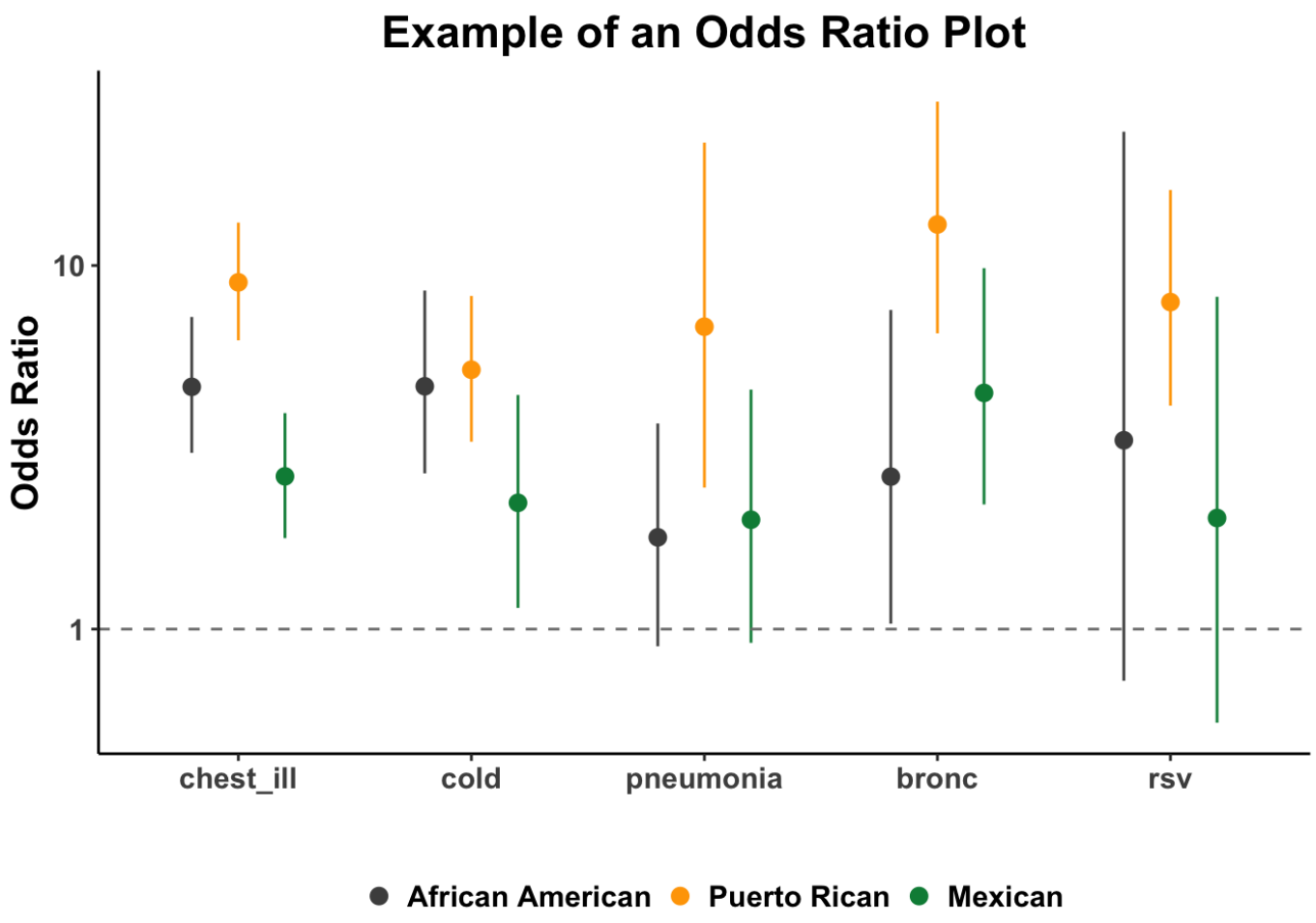
OR	LL	UL	N	p.value	cond
African American					
4.637	3.053	7.217	827	0.0000000	chest_ill
4.653	2.681	8.534	840	0.0000002	cold
1.788	0.897	3.677	840	0.1033120	pneumonia
2.626	1.035	7.543	840	0.0523905	bronc
3.306	0.721	23.324	840	0.1550318	rsv
Puerto Rican					
8.988	6.225	13.117	1045	0.0000000	chest_ill
5.168	3.277	8.246	1046	0.0000000	cold
6.786	2.450	21.764	1046	0.0004405	pneumonia
12.966	6.507	28.217	1046	0.0000000	bronc
7.931	4.117	16.125	1046	0.0000000	rsv
Mexican					
2.631	1.779	3.927	873	0.0000016	chest_ill
2.225	1.144	4.403	861	0.0193849	cold
1.999	0.917	4.557	861	0.0874304	pneumonia
4.462	2.202	9.833	862	0.0000759	bronc
2.021	0.553	8.202	861	0.2920152	rsv

create the plot

```
# convert to factors for aesthetics
  final.df$cond <- factor(final.df$cond,levels=unique(final.df$cond))
  final.df$group <- factor(final.df$group,levels=unique(final.df$group))
```

convert `cond` and `group` to factors so that ggplot will plot the conditions and the group in the order they appear in the data frame

```
# so points in each group don't overlap with each other
dodge <- position_dodge(width=0.6)

# the plot
ggplot(final.df, aes(color = group)) +
  geom_pointrange(aes(x = cond, y = as.numeric(as.character(OR)), ymin = as.numeri
c(as.character(LL)), ymax = as.numeric(as.character(UL))),
                  position = dodge) +
  labs(x = '', y='Odds Ratio') +
  scale_y_log10() +
  theme_classic() +
  geom_hline(aes(yintercept = 1), col = 'gray50', lty = 2) +
  theme(legend.position = "bottom", text = element_text(size=14, face='bold'), plo
t.title = element_text(hjust = 0.5)) +
  guides(color = guide_legend(title = "",override.aes = list(linetype = 0))) +
  scale_color_manual(values = c("grey30","orange","springgreen4")) +
  ggtitle("Example of an Odds Ratio Plot")
```

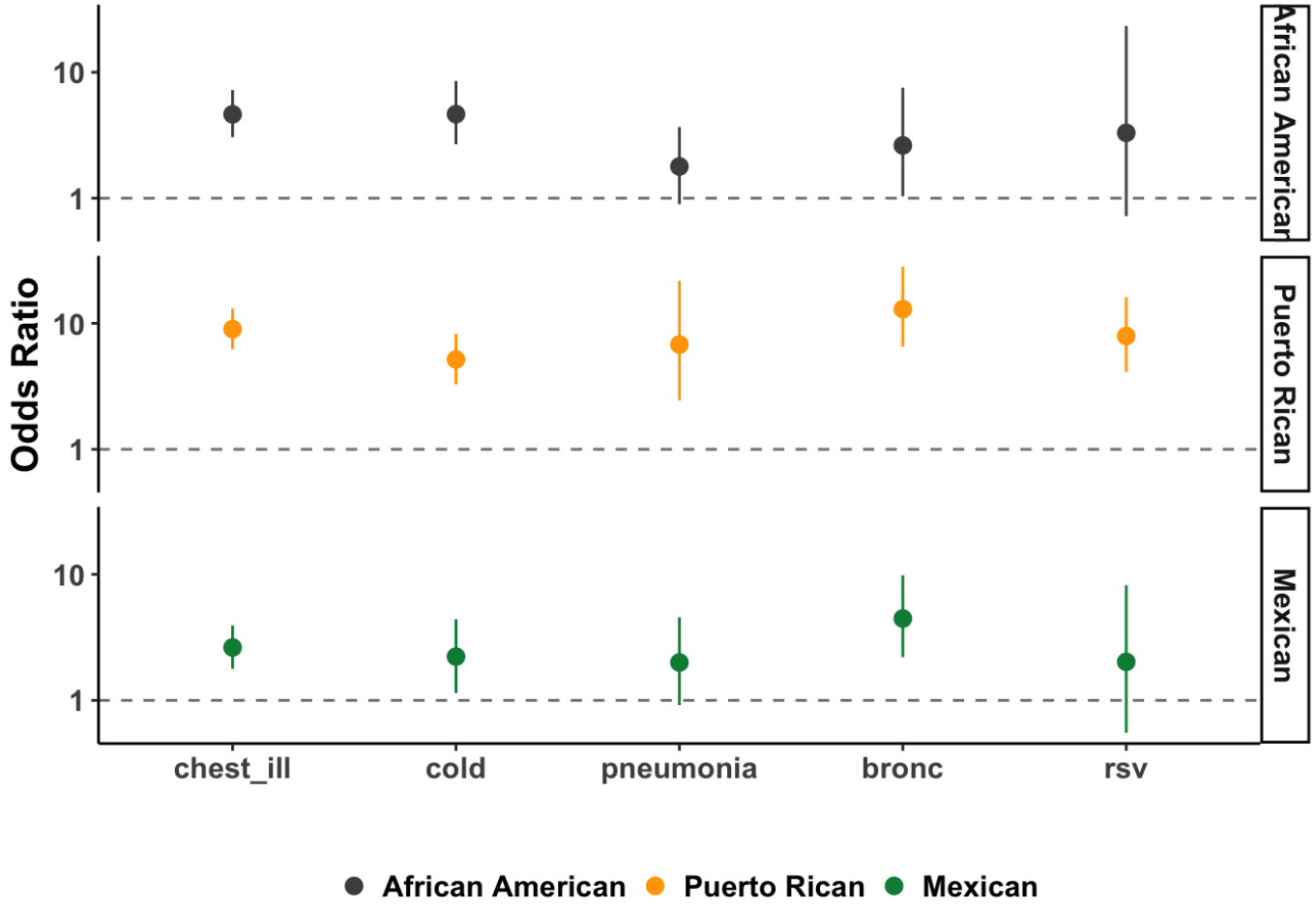


faceting

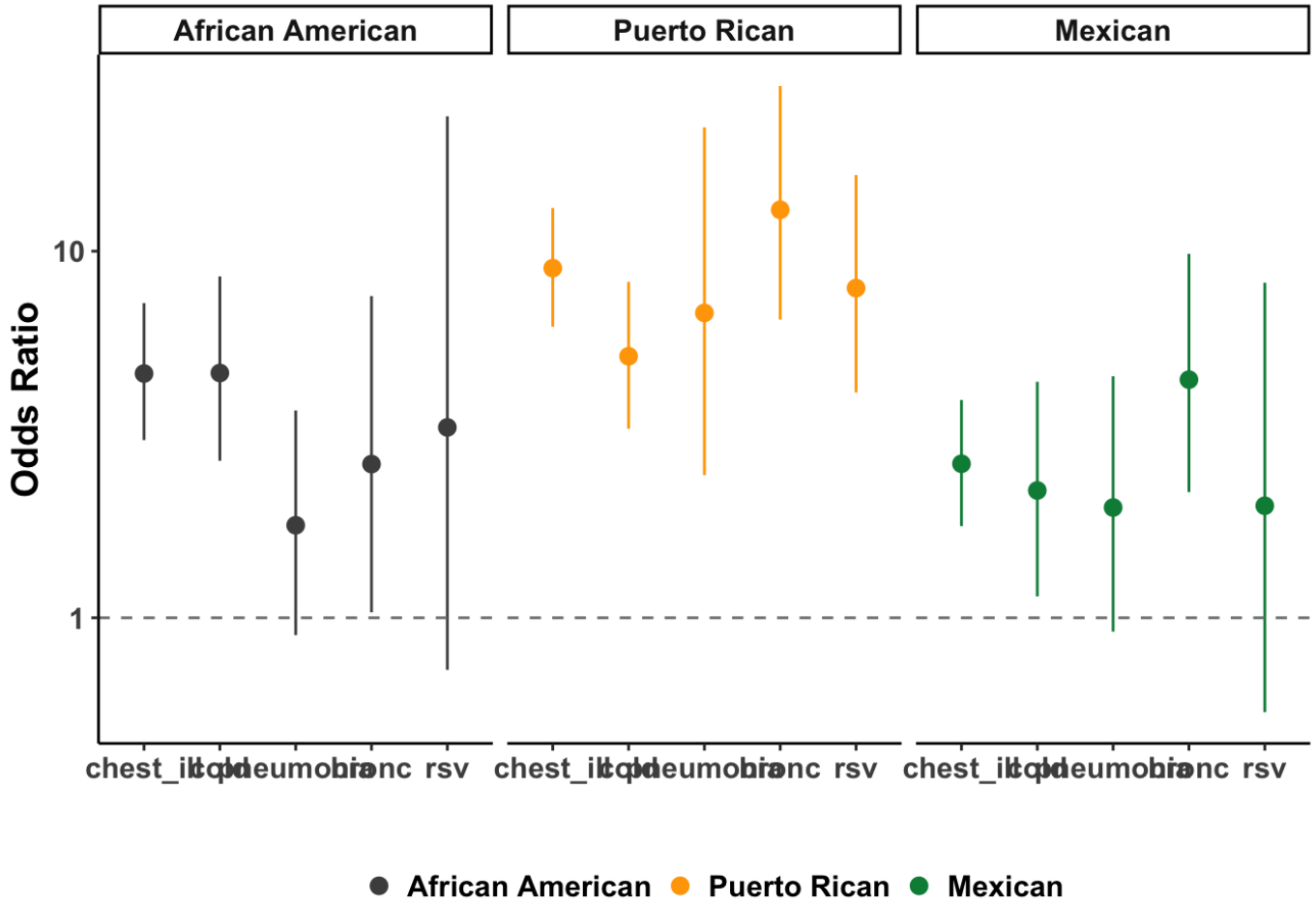
Faceting allows you to seperate your data and plot everything on the same scale. It is useful if you have used up all the different aesthetics (`color` , `alpha` , `size` , `shape` , `fill` , `linetype` , `group`). For this example, I show

1. that you can facet along the x or y axis
2. that you can facet by two groups
3. how to fix common issues that come with faceting

```
# set your base graph
p <- ggplot(final.df, aes(color = group)) +
  geom_pointrange(aes(x = cond, y = as.numeric(as.character(OR)), ymin = as.numeri
c(as.character(LL)), ymax = as.numeric(as.character(UL))),
                  position = dodge) +
  labs(x = '', y='Odds Ratio') +
  scale_y_log10() +
  theme_classic() +
  geom_hline(aes(yintercept = 1), col = 'gray50', lty = 2) +
  guides(color = guide_legend(title = "",override.aes = list(linetype = 0))) +
  theme(legend.position = "bottom", text = element_text(size=14, face='bold')) +
  scale_color_manual(values = c("grey30","orange","springgreen4"))

# facet by group on the y axis
p + facet_grid(group ~ .)
```



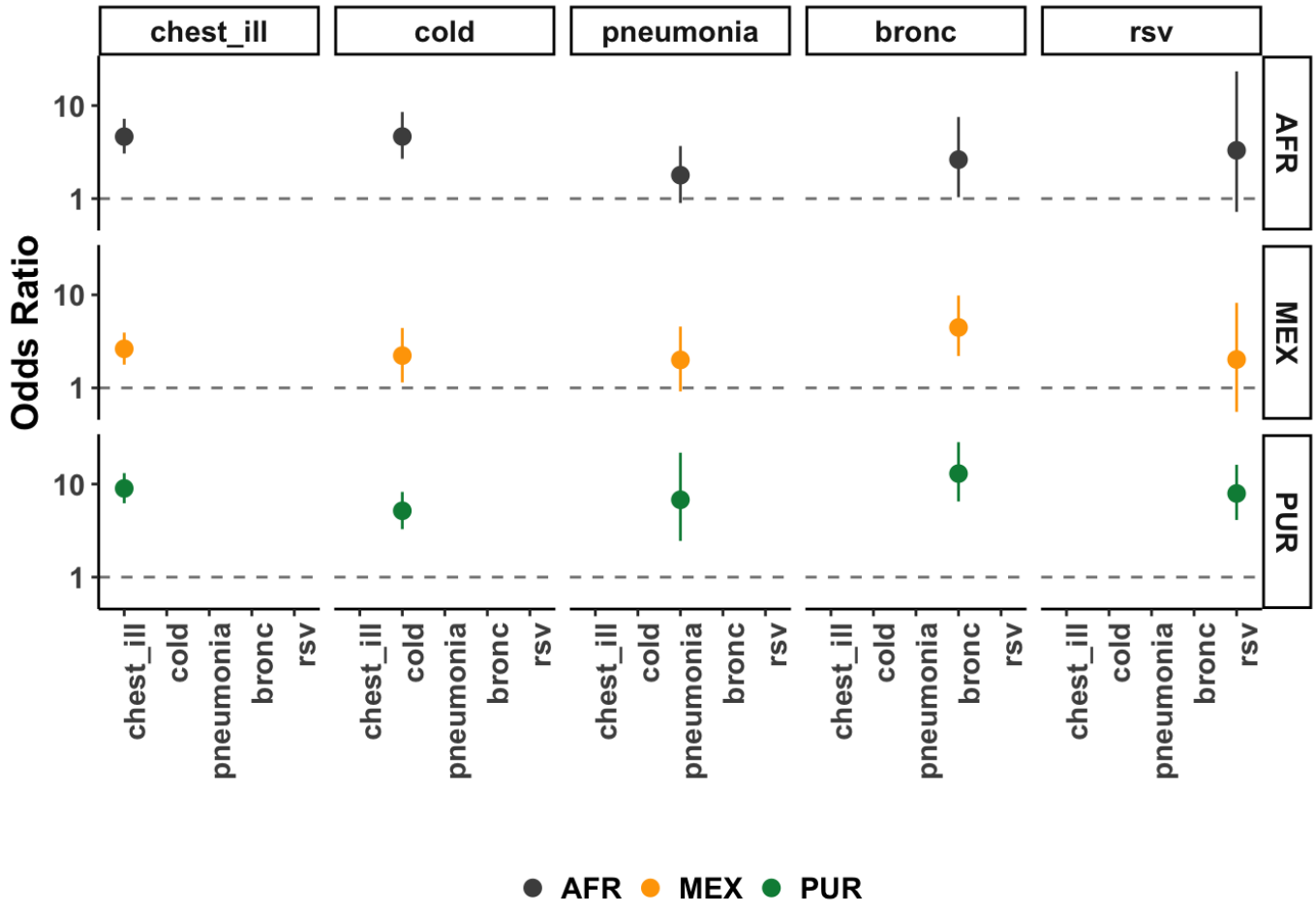

```
# facet by group on the x axis
p+ facet_grid(. ~ group)
```



common issues with faceting

1. facet labels dont fit
2. axis ticks overlap each other
   To fix (1) can be challenging so I like to create new columns in my dataset with shortened label names and use those to build the plot. To fix (2) is as easy as adjsuting the the angle of the text using the `theme` function in ggplot

```
# facet lables aren't easy to fix so lets create shortened labels that are still in
formative
final.df$group.abrv <- NA
final.df$group.abrv <- ifelse(final.df$group == "African American", "AFR", final.df
$group.abrv)
final.df$group.abrv <- ifelse(final.df$group == "Puerto Rican", "PUR", final.df$gro
up.abrv)
final.df$group.abrv <- ifelse(final.df$group == "Mexican", "MEX", final.df$group.ab
rv)

ggplot(final.df, aes(color = group.abrv)) +
  geom_pointrange(aes(x = cond, y = as.numeric(as.character(OR)), ymin = as.numeri
c(as.character(LL)), ymax = as.numeric(as.character(UL))),
                  position = dodge) +
  labs(x = '', y='Odds Ratio') +
  scale_y_log10() +
  theme_classic() +
  geom_hline(aes(yintercept = 1), col = 'gray50', lty = 2) +
  guides(color = guide_legend(title = "",override.aes = list(linetype = 0))) +
  theme(legend.position = "bottom", text = element_text(size=14, face='bold')) +
  scale_color_manual(values = c("grey30","orange","springgreen4")) +
  facet_grid(group.abrv ~ cond) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Muliplotting

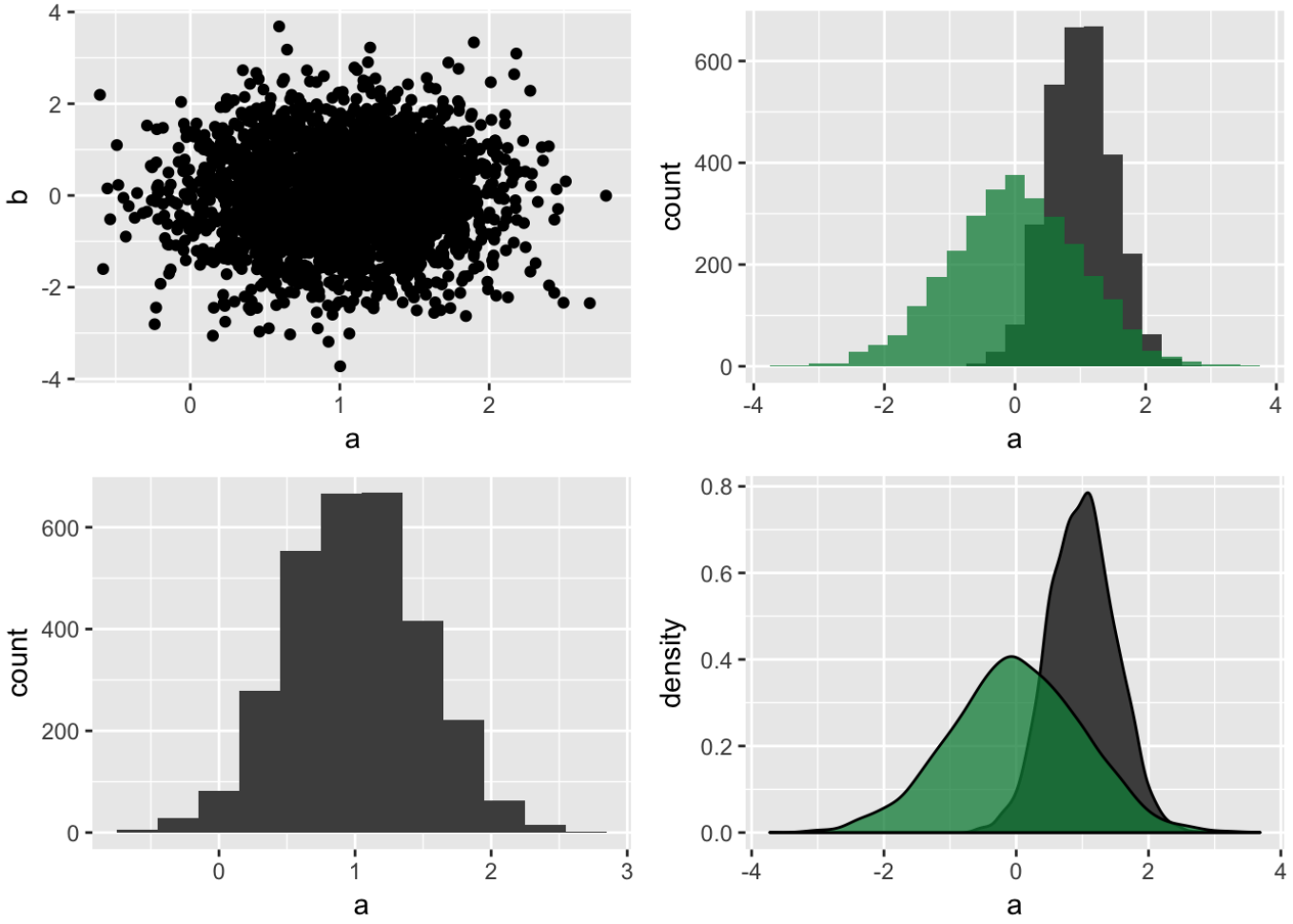
Multiplotting is a great way to place more than one plot in a graphing area. You can have as many as you

like, but 4 is the recommended number.

```
# multiplot function
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

Lets put our multiplot function to the test

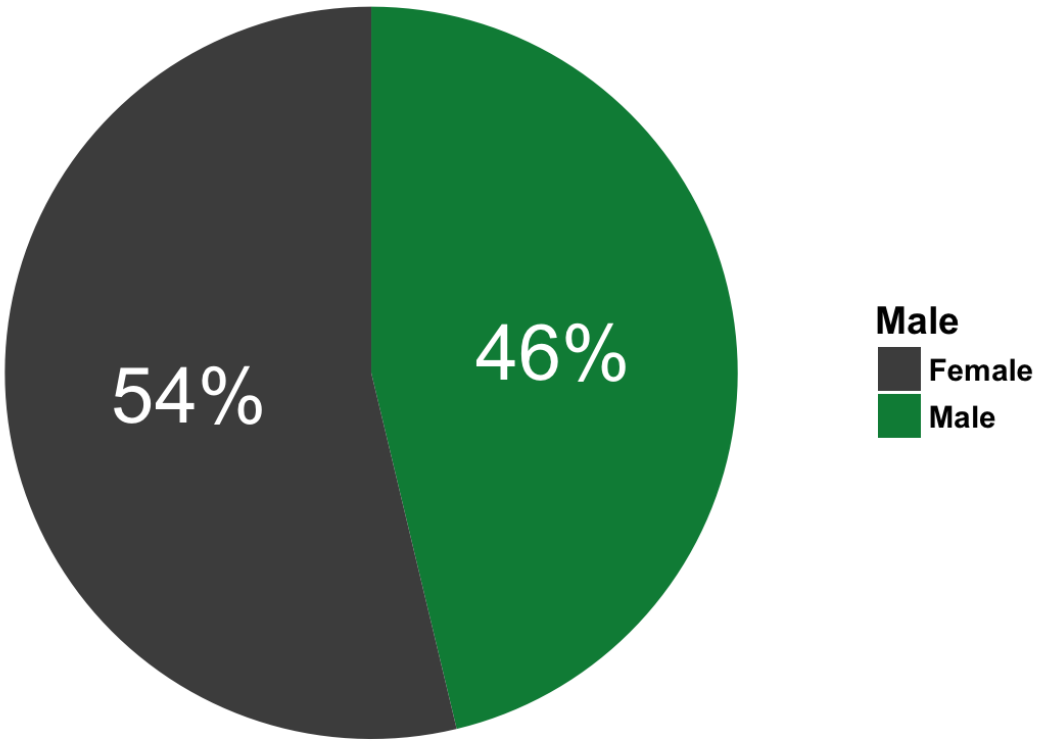
```
a <- rnorm(3000, mean = 1, sd = 0.5)
b <- rnorm(3000, mean = 0, sd = 1)

ex1 <- data.frame(a,b)

p <- ggplot(ex1)
p1 <- p + geom_point(aes(x = a, y = b))
p2 <- p + geom_histogram(aes(a), fill = "grey30", binwidth = 0.3)
p3 <- p + geom_histogram(aes(a), fill = "grey30",  binwidth = 0.3) + geom_histogra
m(aes(b), alpha = 0.75, fill = "springgreen4",  binwidth = 0.3)
p4 <- p + geom_density(aes(a), fill = "grey30") + geom_density(aes(b), alpha = 0.7
5, fill = "springgreen4")

multiplot(p1,p2,p3,p4, cols = 2)
```



Pie Charts

Pie charts are just like bar plots except on polar corrdinates

```
data.check <- mydata
data.check$Male <- reorder(data.check$Male, X = data.check$Male, FUN = function(x)
-length(x))
# the at calculation is finding the centers of the wedges. (It's easier to think of
them as the centers of bars in a stacked bar plot
at <- as.numeric(cumsum(sort(table(data.check$Male)))-0.5*sort(table(data.check$Mal
e)))
label=paste0(round(sort(table(data.check$Male))/sum(table(data.check$Male)),2) * 1
00,"%")


ggplot(data.check,aes(x="", fill = Male)) +
  geom_bar(width = 1) +
  coord_polar(theta="y") +
  annotate(geom = "text", y = at, x = 1, label = label, color  = "white", size=10)
+
  theme_void() +
  theme(text = element_text(size=14, face='bold'), plot.title = element_text(hjust
= 0.5)) +
  scale_fill_manual(values = c("grey30","springgreen4")) +
  ggtitle("Example of a Pie Chart")
```

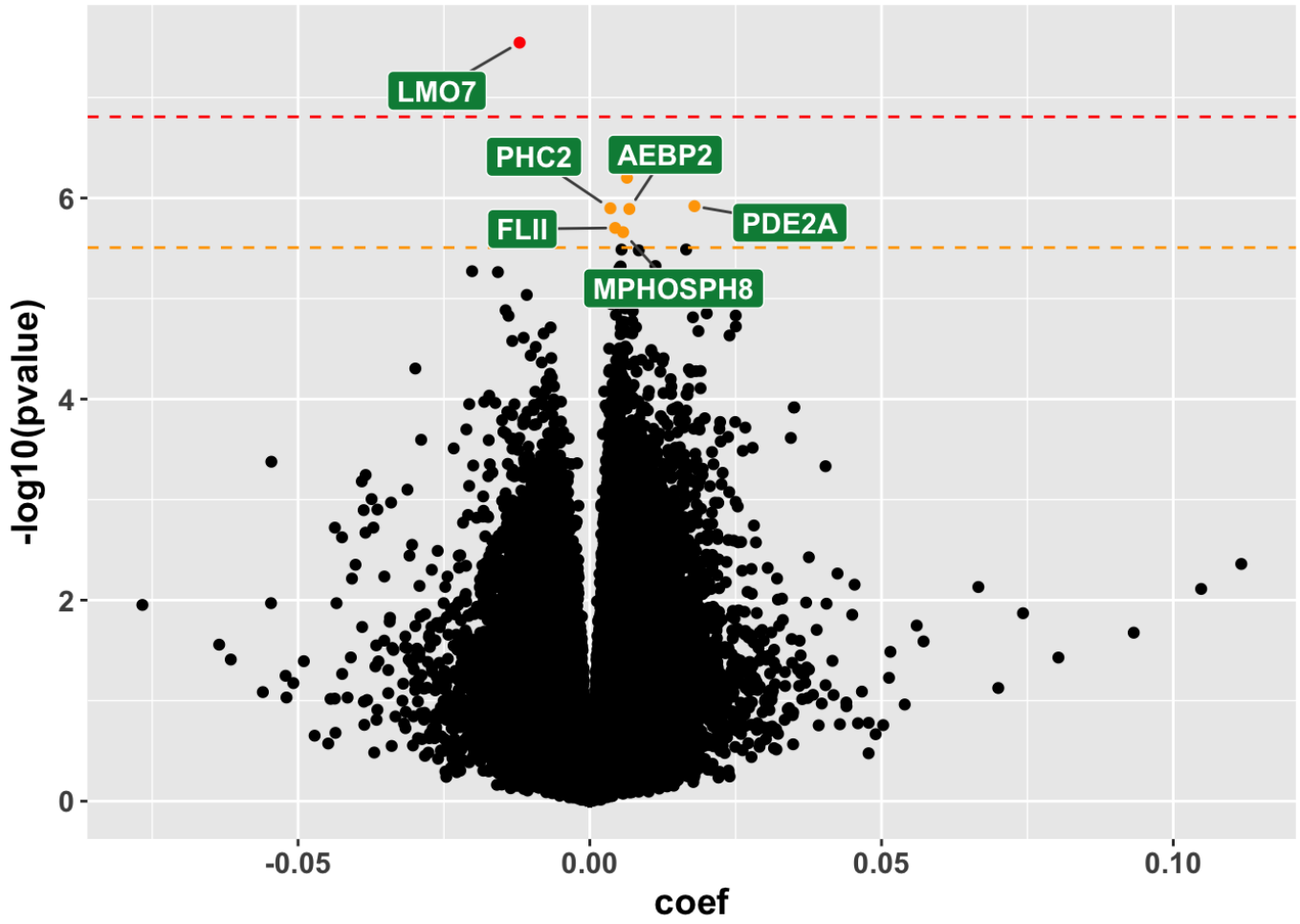


volcano plots

```
# this example will use methylation data
  cold <- read.csv("~/Dropbox/Viral_meth_SO/cold_illness_methylation.csv", header =
T, fill = T, na.strings = "")

# use a regular expression to reduce the gene names from gene;gene;gene;gene etc. t
o just gene\
  n <- nrow(cold)
  bonf <- 0.05/n
  sugg <- 1/n
  cold$padj <- cold$pvalue/sugg
  cold$plot.colors <- ifelse(cold$padj <= 1, "orange","black")
  cold$plot.colors <- ifelse(cold$padj <= 0.05, "red",cold$plot.colors)
  cold$simple.gene.name <- gsub("([A-Z0-9]+)((;[A-Z0-9]+)+)?","\\1", as.character(c
old$UCSC_REFGENE_NAME))
  cold$gene_name <- ifelse(cold$padj <= 1, as.character(cold$simple.gene.name),NA)
# only significant genes get gene_name so as to not overwhelm the volcano plot

  # check our the data we are using to create the plot
  str(cold)
```

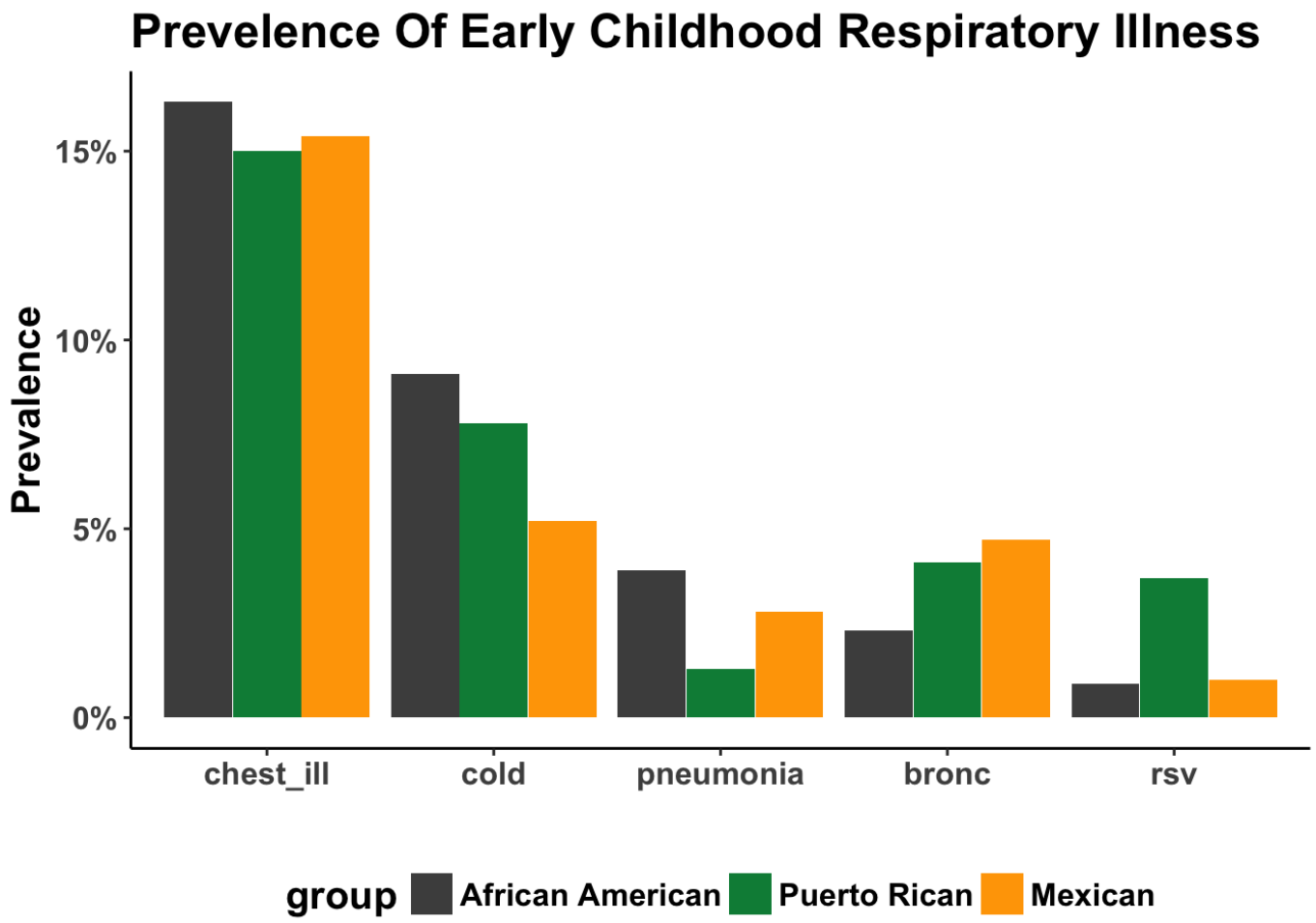
```
FALSE 'data.frame': 321503 obs. of  13 variables:
FALSE  $ X                          : Factor w/ 321503 levels "cg00000029","cg0000
0236",..: 1 2 3 4 5 6 7 8 9 10 ...
FALSE  $ coef                       : num  -2.62e-03 -2.36e-03 -6.66e-03 -1.62e-03
-9.35e-05 ...
FALSE  $ se                         : num  0.00503 0.00319 0.00611 0.00473 0.00343
...
FALSE  $ pvalue                     : num  0.602 0.459 0.276 0.733 0.978 ...
FALSE  $ UCSC_REFGENE_NAME          : Factor w/ 36604 levels "A1BG","A1CF;A1CF;A1CF
",..: 26412 34499 412 28538 NA 21614 18634 33480 18163 7160 ...
FALSE  $ UCSC_REFGENE_GROUP         : Factor w/ 3461 levels "1stExon","1stExon;1stE
xon",..: 2458 626 628 2458 NA 3426 1689 1933 1689 1156 ...
FALSE  $ RELATION_TO_UCSC_CPG_ISLAND: Factor w/ 5 levels "Island","N_Shelf",..: 3 N
A 3 5 3 1 1 5 4 1 ...
FALSE  $ CHR                        : int  16 8 14 8 1 15 9 19 6 3 ...
FALSE  $ MAPINFO                    : int  53468112 42263294 69341139 41167802 230
560793 23034447 139997924 54695678 25282779 128902377 ...
FALSE  $ padj                       : num  193626 147685 88583 235544 314525 ...
FALSE  $ plot.colors                : chr  "black" "black" "black" "black" ...
FALSE  $ simple.gene.name           : chr  "RBL2" "VDAC3" "ACTN1" "SFRP1" ...
FALSE  $ gene_name                  : chr  NA NA NA NA ...
```

```
# create an awesome volcano plot
# to avoid overlapping gene names, use geom_label_repeal


  ggplot() +
    geom_point(data = cold,
               aes(x=coef, y = -log10(pvalue)),
               colour = cold$plot.colors) +
    # bonferroni significance threshold
    geom_hline(aes(yintercept=-log(bonf/1, 10)),
               colour="red",
               linetype="dashed") +
    # suggestive significance threshold line
    geom_hline(aes(yintercept=-log(bonf/0.05, 10)),
               colour="orange",
               linetype="dashed") +
    # non overlapping gene labels
    geom_label_repel(aes(x=cold$coef, y = -log10(cold$pvalue), label = cold$gene_n
ame),
                     fontface = 'bold',
                     fill = "springgreen4",
                     color = "white",
                     box.padding = 0.35,
                     point.padding = 0.5,
                     segment.color = 'grey30') +
    theme(legend.position = "none", text = element_text(size=14, face='bold'))
```



Prevelence plots

```
prev.final <- data.frame()
for (i in illness) {
  tab <- CrossTable(x = mydata$group, y = mydata[,i], format = 'SAS', prop.r = T,
prop.c = F, prop.t = F, prop.chisq = F, digits = 1)
  prev.tab <- data.frame()
  for (j in tab$prop.row[,2]) {
    prev.tab<-rbind(prev.tab,cbind(i,round(j,digits = 3)))
  }
  names(prev.tab) = c('cond', 'prev')
  prev.tab$group <- c('African American','Mexican','Other Latino','Puerto Rican')
  prev.final <- rbind(prev.final, prev.tab)
}

tab.sort <- prev.final[order(prev.final$group),]
tab.sort2 <- tab.sort[c(1:5,11:20,6:10),]
prev.tab.clean <- subset(tab.sort2, group != "Other Latino")
# make axis labels legible by shortening them
prev.tab.clean$cond.abv <- NA
prev.tab.clean$cond.abv <- ifelse(prev.tab.clean$cond == "chest_ill_before2yr", "ch
est_ill", prev.tab.clean$cond.abv)
prev.tab.clean$cond.abv <- ifelse(prev.tab.clean$cond == "cold_before2yr", "cold",
prev.tab.clean$cond.abv)
prev.tab.clean$cond.abv <- ifelse(prev.tab.clean$cond == "pneumonia_before2yr", "pn
eumonia", prev.tab.clean$cond.abv)
prev.tab.clean$cond.abv <- ifelse(prev.tab.clean$cond == "bronchitis_before2yr", "b
ronc", prev.tab.clean$cond.abv)
prev.tab.clean$cond.abv <- ifelse(prev.tab.clean$cond == "bronchiolitis_RSV_before2
yr", "rsv", prev.tab.clean$cond.abv)


prev.tab.clean$cond.abv <- factor(prev.tab.clean$cond.abv,levels=unique(prev.tab.cl
ean$cond.abv))
prev.tab.clean$group <- factor(prev.tab.clean$group,levels=unique(prev.tab.clean$gr
oup))

# plot
ggplot(prev.tab.clean, aes(x=cond.abv, y=as.numeric(as.character(prev)), fill=group
)) +
  geom_bar(stat='identity', position = position_dodge(width=0.91)) +
  labs(x = '', y='Prevalence') +
  scale_y_continuous(labels=percent) +
  ggtitle("Prevelence Of Early Childhood Respiratory Illness") +
  theme_classic() +
  theme(legend.position="bottom", text = element_text(size=15, face='bold')) +
  scale_fill_manual(values = c("grey30", "springgreen4", "orange"))
```



Exporting a plot

```
pdf('~/file/path/to/export.pdf', width = 20, height = 8)
### your plot ###
dev.off()
```