Lab: ggplot

36-600

Fall 2022

Data

We'll importing the same data that we used for the `dplyr` lab, with rows with data with value 99 (i.e., rows with missing data) removed:

```
suppressMessages(library(tidyverse))
rm(list=ls())
file.path = "https://raw.githubusercontent.com/pefreeman/36-290/master/EXAMPLE_DATASETS/
BUZZARD/Buzzard_DC1.Rdata"
load(url(file.path))
set.seed(101)
s = sample(nrow(df),4000)
df = df[s,]
df %>% filter(.,u!=99&g!=99&r!=99&i!=99&z!=99&y!=99) -> df.new
predictors = df.new[,-c(7:14)]
redshift   = as.vector(df.new[,14])

type = rep("FAINT",nrow(predictors))
w = which(predictors$i<25)
type[w] = "BRIGHT"
type = factor(type)
predictors = cbind(type,predictors)

df = cbind(predictors,redshift)
```

If everything loaded correctly, you should see a variable in your global environment called `df`, a data frame with 3607 rows and 8 columns.

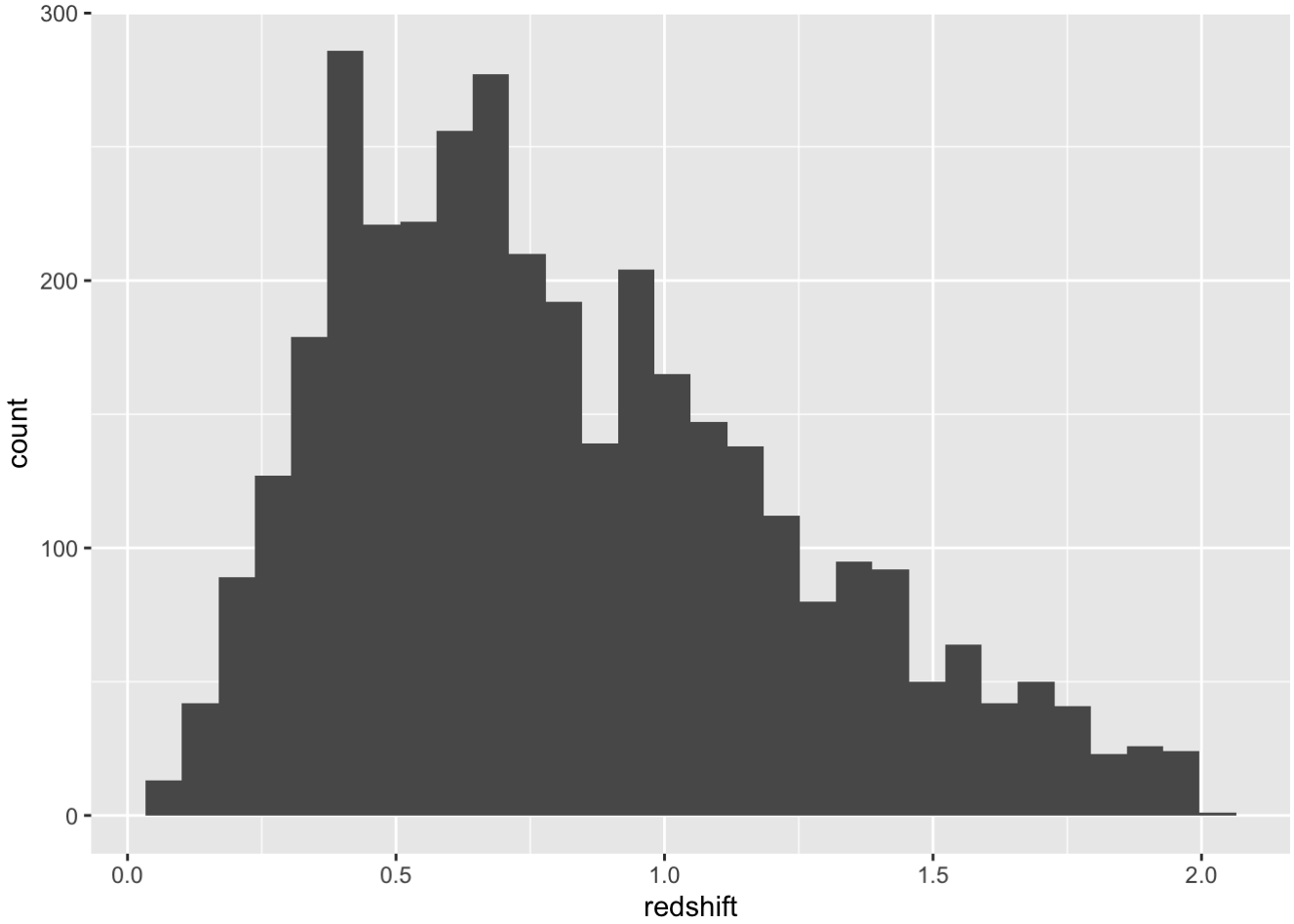
ggplot

We will use `ggplot` to create our plots. In order to become comfortable quickly with `ggplot`, you should read through Chapter 3 (online; Chapter 1 in print) of *R for Data Science* by Wickham and Grolemund, available for free here (http://r4ds.had.co.nz/).

The following is an example of a `ggplot` call:

```
ggplot(data=df,mapping=aes(x=redshift)) +
  geom_histogram() # ignore the picky complaint about bins
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



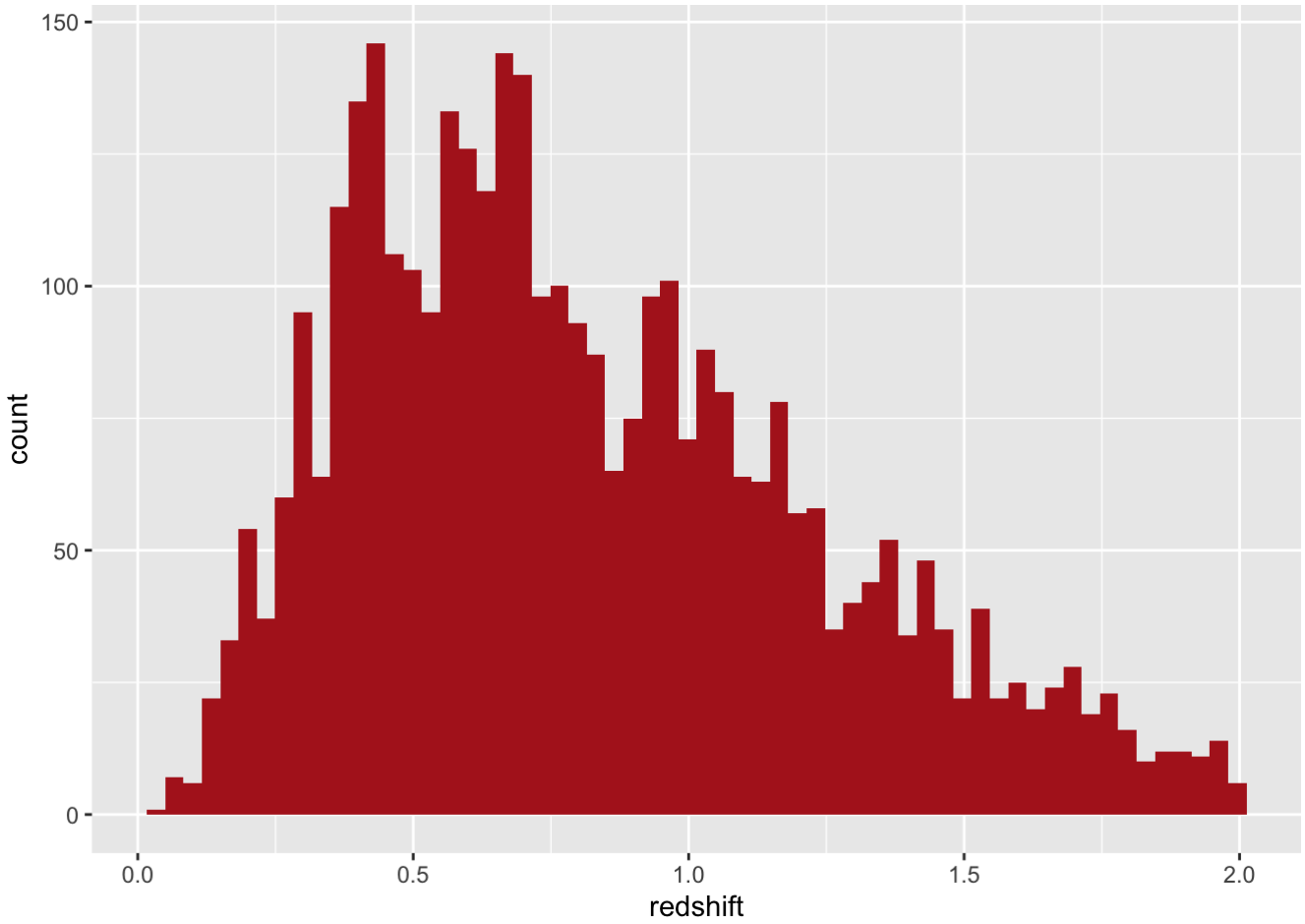
Note the structure here: the first function is `ggplot()`, and you pass a data frame (as `data`) and an identification of which feature goes along which axis (as `mapping`). Once you've identified the data and the mapping, you "add on" another function that describes what you will do to the data; in this case, that function is `geom_histogram()`.

Questions

Question 1

Re-run ggplot from above, setting the number of bins to be 60. (Use the documentation for `geom_histogram()` to figure out how to do this…e.g., type `?geom_histogram` in the Console pane.) Also, change the color of the histogram via the `fill` argument. Google "R colors" to find listings of the names of `R`'s colors. My personal favorite is "papayawhip".

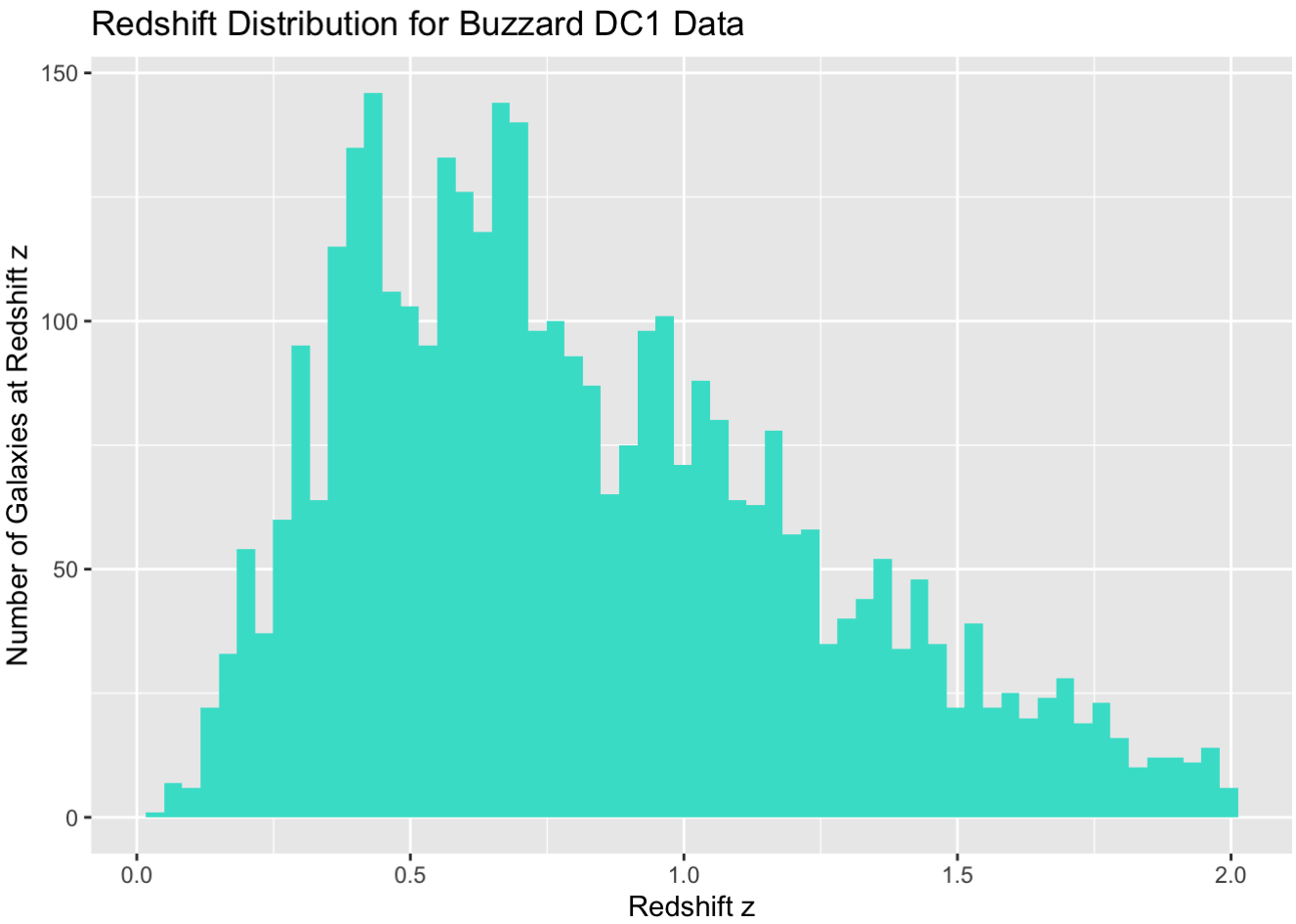
```
ggplot(data=df,mapping=aes(x=redshift)) +
  geom_histogram(bins=60,fill="firebrick")
```



Question 2

Repeat Question 1, but change the x-axis label to “Redshift z” and the y-axis label to “Number of Galaxies at Redshift z”. Also, add a title to the plot: “Redshift Distribution for Buzzard DC1 Data”. Google phrases like “how to add a title to ggplot”. Trust me: Googling is the fastest way to figure out what function does what in the `ggplot` world.

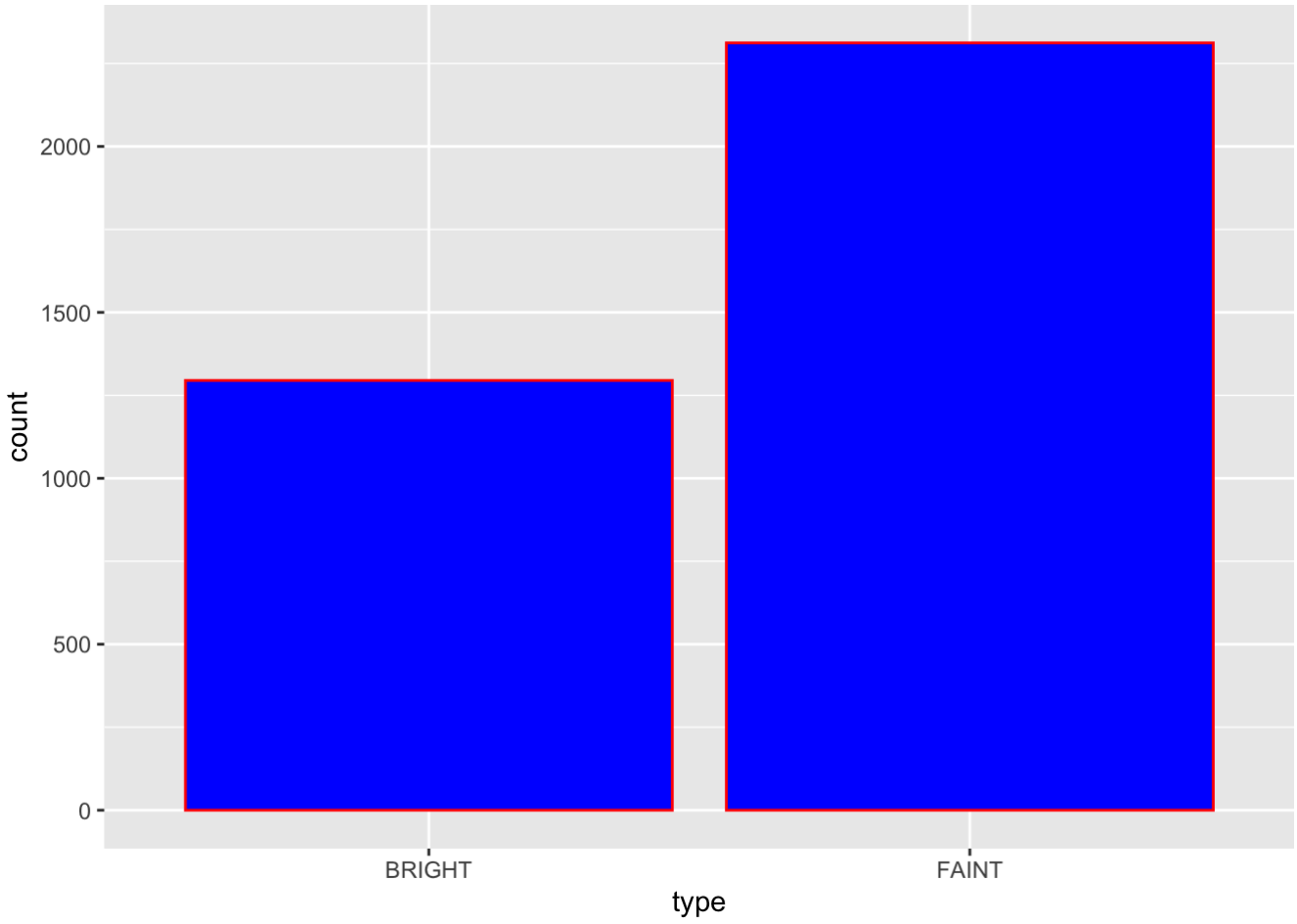
```
ggplot(data=df,mapping=aes(x=redshift)) +
  geom_histogram(bins=60,fill="turquoise") +
  xlab("Redshift z") +
  ylab("Number of Galaxies at Redshift z") +
  ggtitle("Redshift Distribution for Buzzard DC1 Data")
```



Question 3

Construct a bar chart that shows the distribution of galaxy `type`, here defined to be `BRIGHT` and `FAINT`. Remember that a bar chart is a representation of a probability mass function. As a reminder to yourself, also show the number of counts for each type, coded in some manner that you've learned thus far.

```
ggplot(data=df,mapping=aes(x=type)) +
  geom_bar(color="red",fill="blue")
```




```
table(df$type)  # or...
```

```
##
## BRIGHT  FAINT
##   1295   2312
```

```
df %>% group_by(.,type) %>% summarize(.,Number=n())  # or...
```

```
## # A tibble: 2 × 2
##   type   Number
##   <fct>   <int>
## 1 BRIGHT   1295
## 2 FAINT    2312
```

```
df %>% group_by(.,type) %>% tally(.)  # or...
```

```
## # A tibble: 2 × 2
##   type       n
##   <fct>  <int>
## 1 BRIGHT  1295
## 2 FAINT   2312
```

```
df %>% count(.,type)
```

```
##     type    n
## 1 BRIGHT 1295
## 2  FAINT 2312
```

Question 4

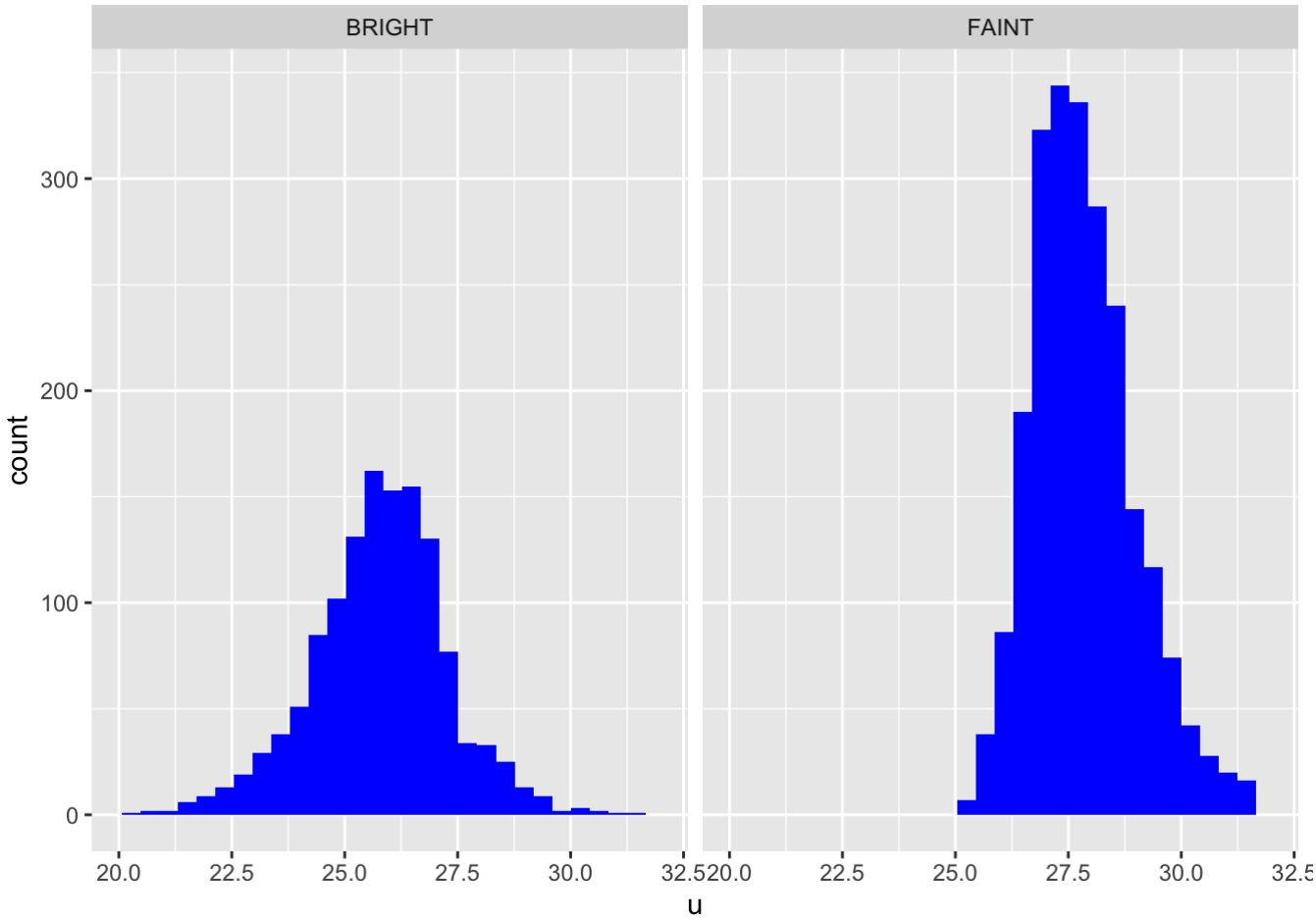
Utilize facet wrapping to display two histograms side-by-side: one showing `u` -band magnitudes for `BRIGHT` data, and one showing `u` -band magnitudes for `FAINT` data. Note that to compare distributions, it is helpful *to have the data span the same scale*. So add a function to the string of functions that sets the x-axis limits to be 20 and 32.

```
ggplot(data=df,mapping=aes(x=u)) +
  geom_histogram(fill="blue") +
  facet_wrap(~type) +
  xlim(20,32)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 20 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```

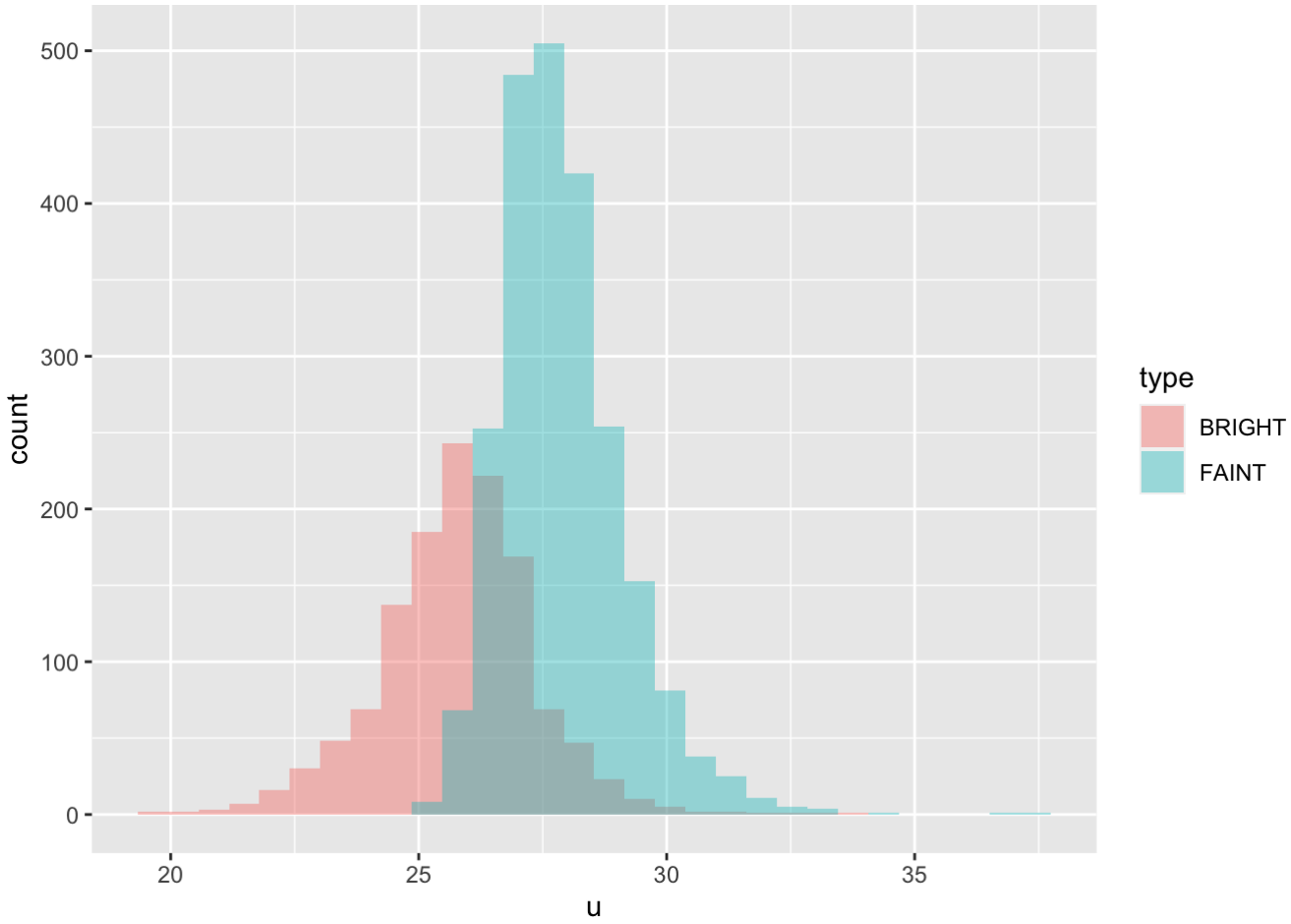


Question 5

Show the same information as in Question 4, except use overlapping, partially transparent histograms. Construct such a histogram (again for `u`). (Hints: in the call to `aes()` in the `ggplot()` call, include `fill=type`, and in the `geom_histogram()` call, include two new arguments: `alpha` and `position="Identity"`. The `alpha` argument controls the transparency: 0 for invisible to 1 for totally opaque.

```
ggplot(data=df,mapping=aes(x=u,fill=type)) +
  geom_histogram(alpha=0.4,position="Identity")
```

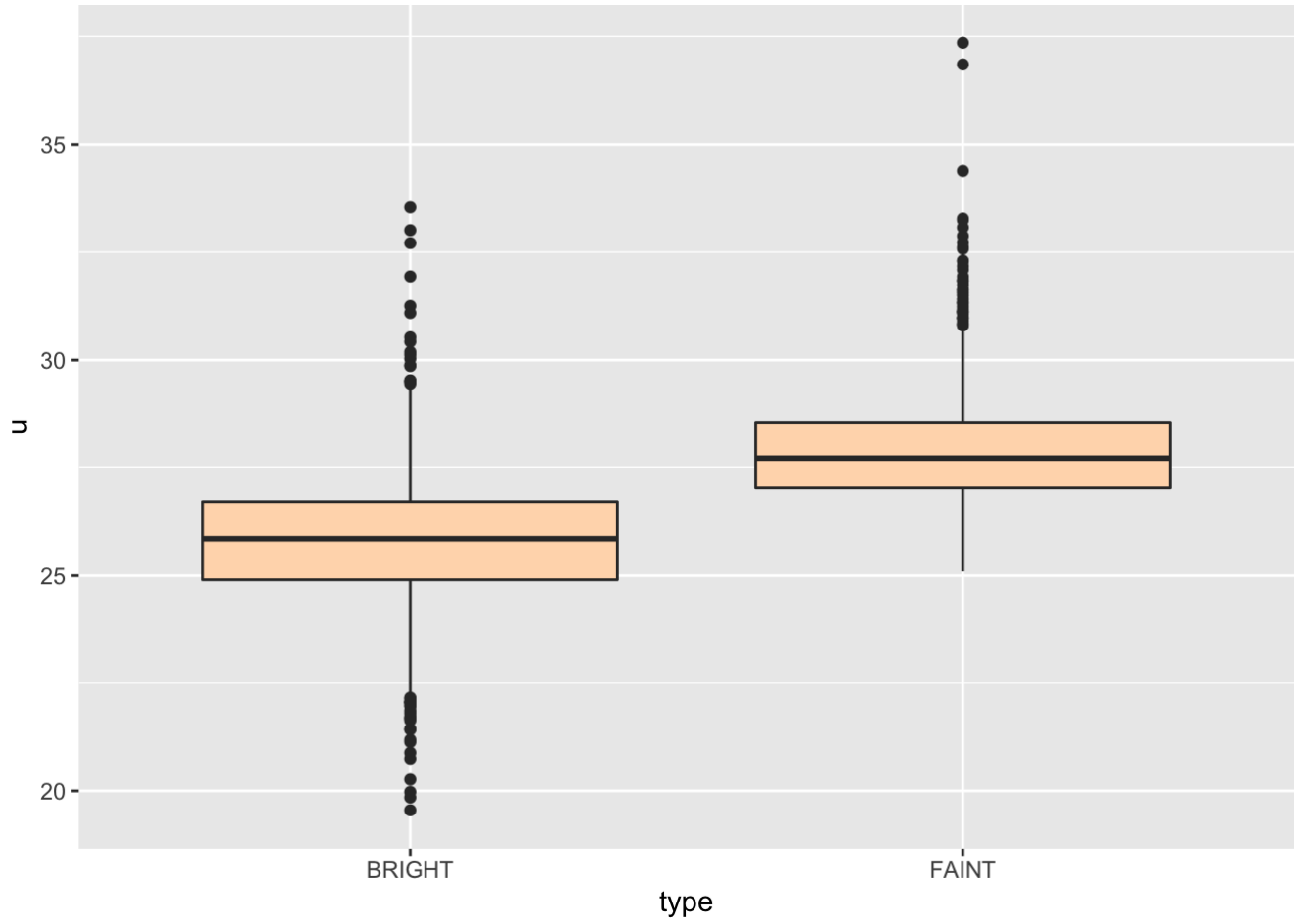
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Question 6

Show the same information as in Question 4, except that instead of using side-by-side histograms, use side-by-side boxplots. Unlike in Question 4, you do not need to apply `facet_wrap()`. Instead, for the aesthetics, specify that `x` is `type` and `y` is `u`.

```
ggplot(data=df,mapping=aes(x=type,y=u)) +
  geom_boxplot(fill="peachpuff")
```



Question 7

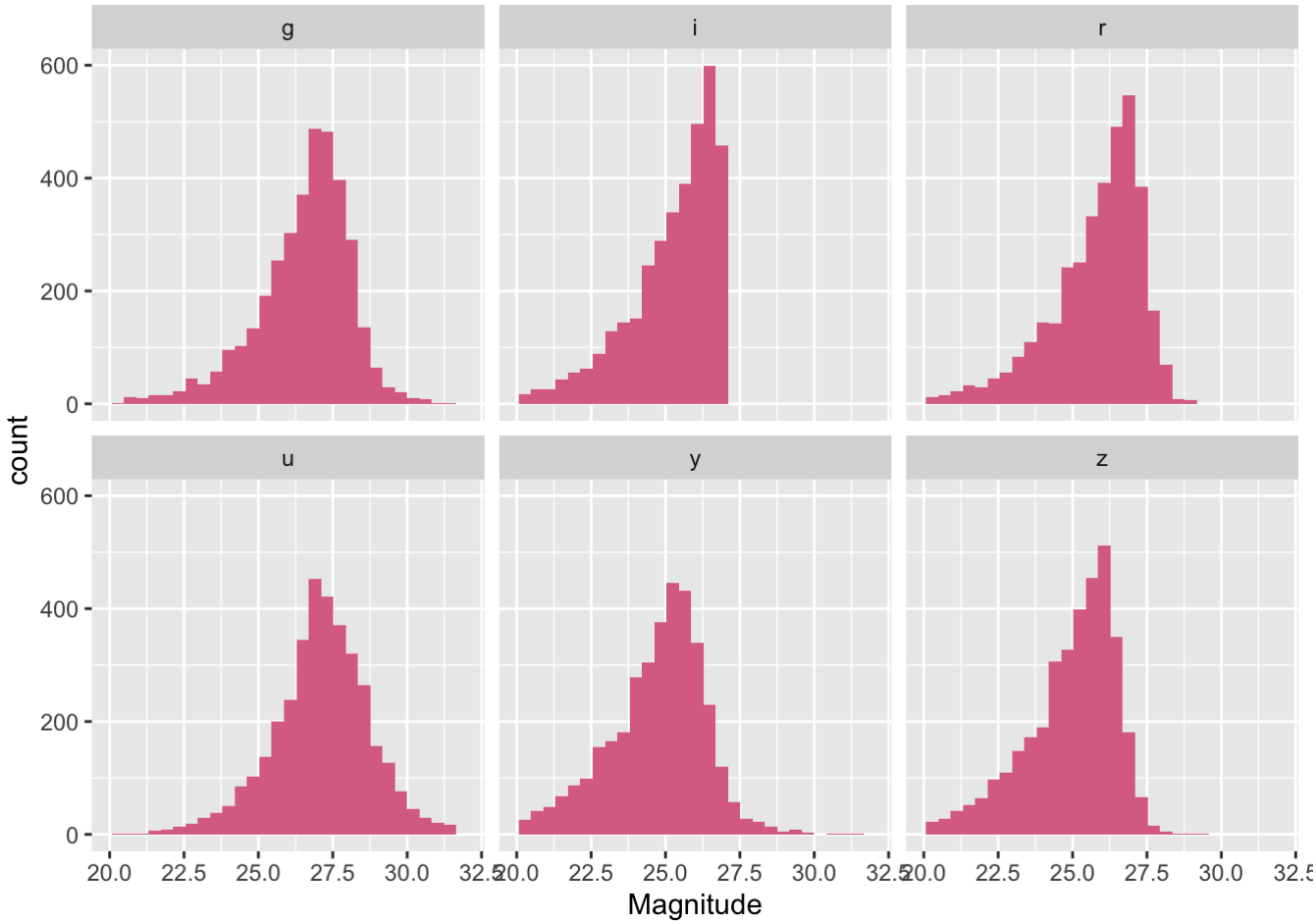
Create a faceted histogram for the `u`, `g`, `r`, `i`, `z`, and `y` columns in the `df` data frame. This means you'll want to pipe `df` to `select()` (or, perhaps, `dplyr::select()`), then to `gather()`, and save the output to a new variable, then you'll want to pass this new data frame into `ggplot()`. Set the x-axis limits to be 20 and 32. Replace the x-axis label with "Magnitude".

```
df %>%
  dplyr::select(.,u,g,r,i,z,y) %>%
  gather(.) -> df.new
ggplot(data=df.new,mapping=aes(x=value)) +
  geom_histogram(fill="palevioletred") +
  facet_wrap(~key) +
  xlim(20,32) +
  xlab("Magnitude")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 230 rows containing non-finite values (stat_bin).
```

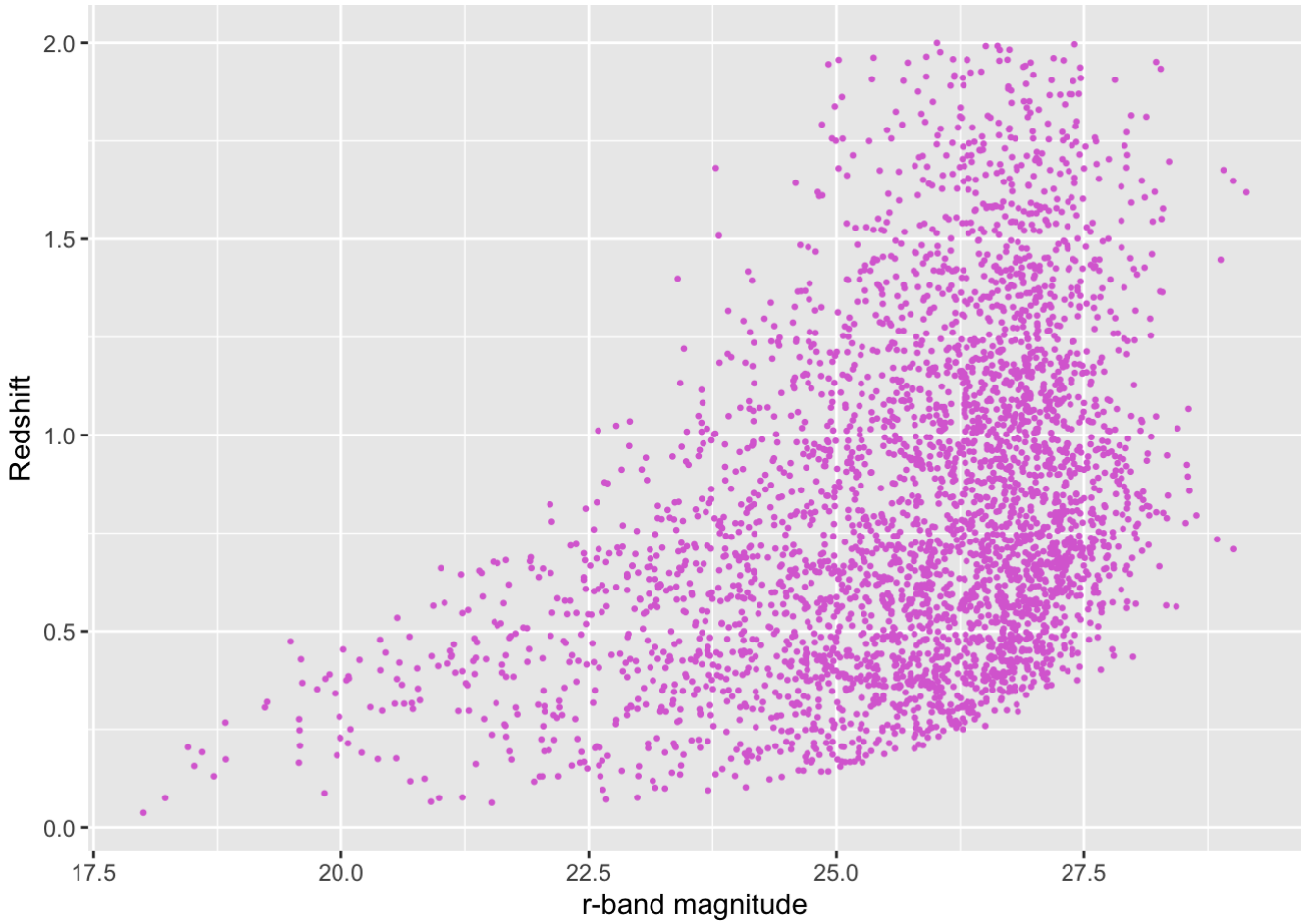
```
## Warning: Removed 12 rows containing missing values (geom_bar).
```



Question 8

Use `geom_point()` to create a scatter plot of `redshift` vs. r-band magnitude. (Remember, one plots y vs. x, so `redshift` will go on the y-axis here.) Make the point size 0.5, change the x-axis label to "r-band magnitude", and change the y-axis label to "Redshift". Oh, and add some color!

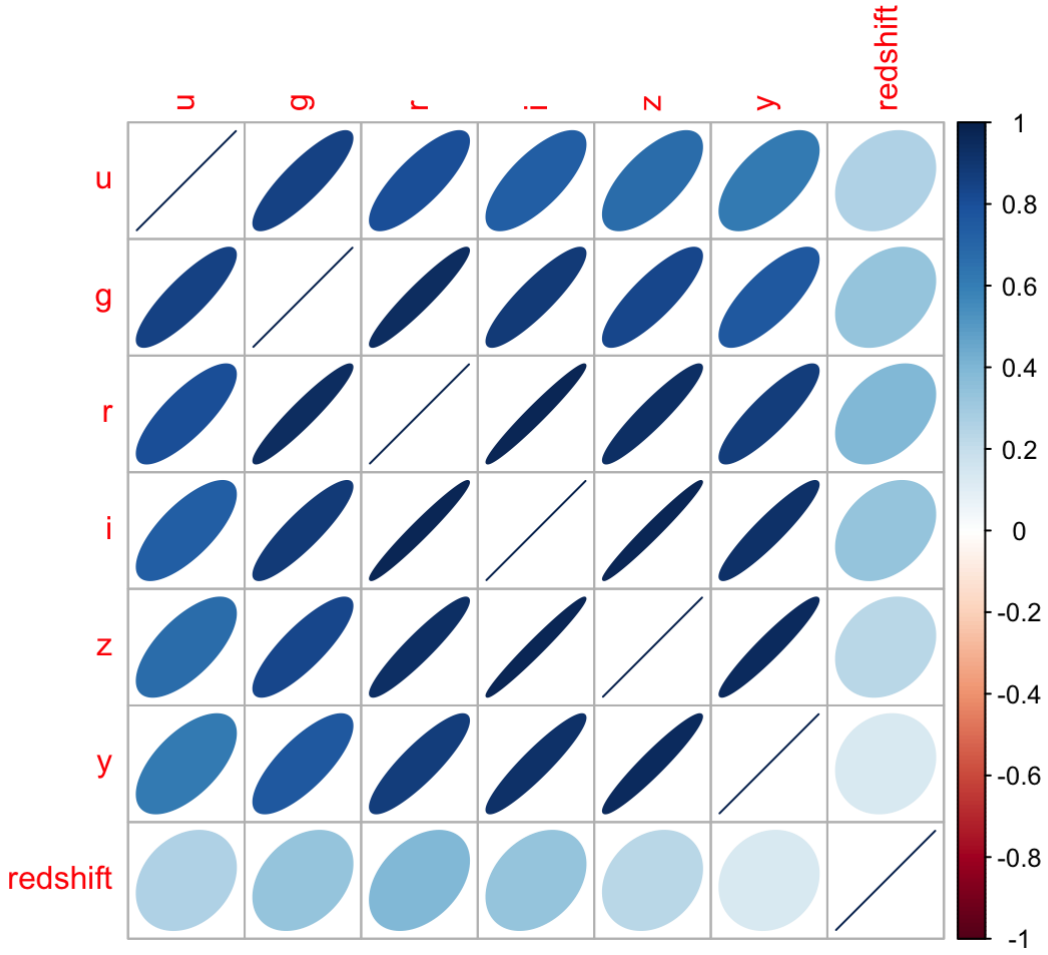
```
ggplot(data=df,mapping=aes(x=r,y=redshift)) +
  geom_point(color="orchid",size=0.5) +
  xlab("r-band magnitude") +
  ylab("Redshift")
```



Question 9

Use the `corrplot()` function to assess the levels of correlation between the different columns of the `df` data frame, except `type`. Utilize the example in the notes. Note that you make need to install the `corrplot` package; you can do so by, e.g., typing `install.packages("corrplot")` in the Console, at the prompt. What should you see? That magnitudes are *very* highly correlated.

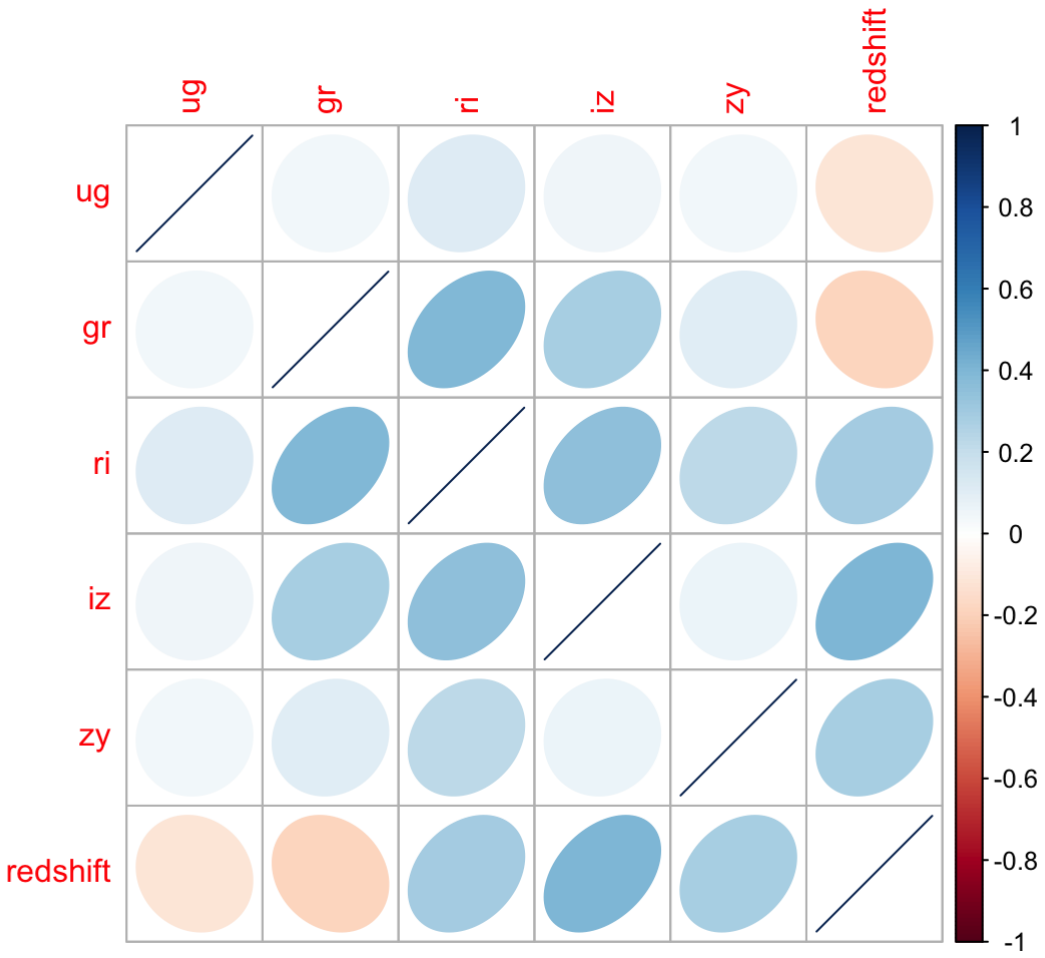
```
library(corrplot)
df %>%
  dplyr::select(.,u,g,r,i,z,y,redshift) %>%
  cor(.) %>%
  corrplot(.,method="ellipse")
```



Question 10

Repeat Question 9, but utilize `mutate()` to create five new variables: `ug`, the difference between `u` and `g`; `gr`, the difference between `g` and `r`; etc. Select only the five new variables, and plot the correlation plot. The new variables are dubbed *colors* and are often used by astronomers because they should only be a function of the physics of galaxies, as opposed to physics and distance. Ultimately, you should find that the correlation between colors is *far less* than the correlation between magnitudes.

```
df %>%
  mutate(.,ug=u-g,gr=g-r,ri=r-i,iz=i-z,zy=z-y) %>%
  dplyr::select(.,ug,gr,ri,iz,zy,redshift) %>%
  cor(.) %>%
  corrplot(.,method="ellipse")
```


ug
gr
ri
iz
zy
redshift
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1