R Ladies Chicago September Meetup ~ ggplot2

Kaelen L. Medeiros

09/28/2017

About me:

- MS in Biostatistics (2016) from LSUHSC
- Data Scientist @ HERE Technologies + freelance health DS
- Learning Japanese, read a lot of books, listen to a lot of podcasts
- Volunteer coordinator for ChickTech Chicago @ChickTechChi
- 100% obsessed with my cat, Scully

Roadmap

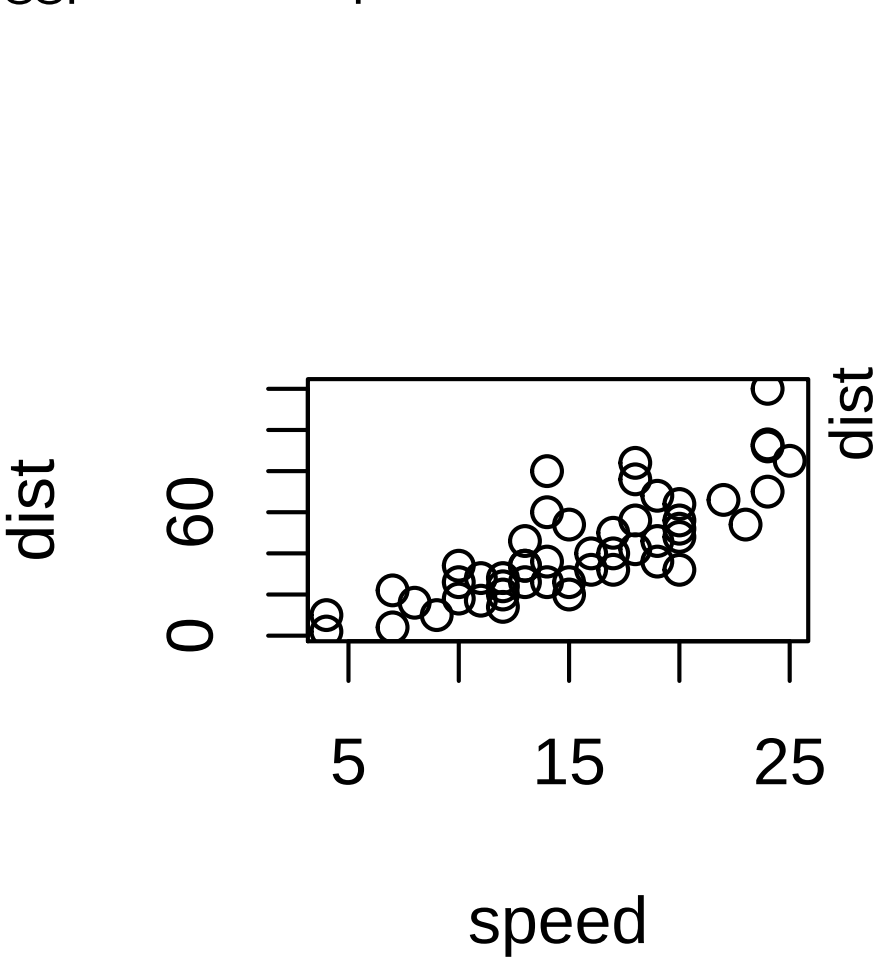
- ggplot2 basics
- Bar charts
- Histograms
- Box plots
- Scatterplots

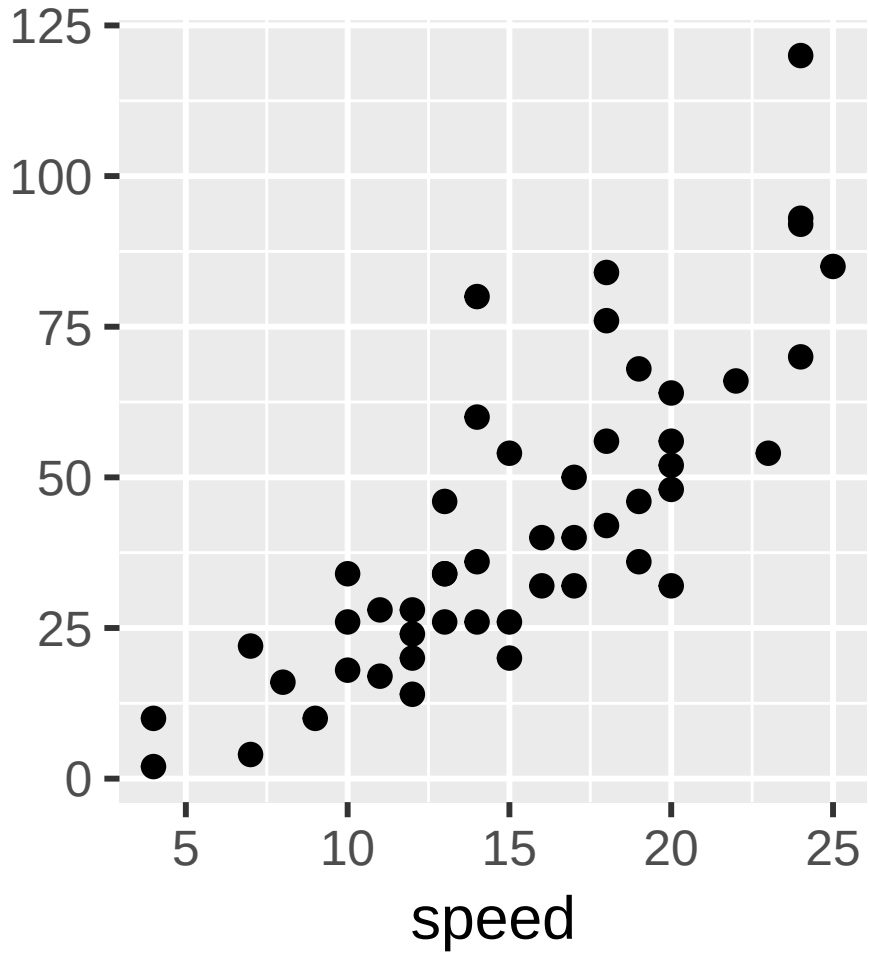
About ggplot2

- Created by Hadley Wickham in 2005
- Based on ~the grammar of graphics~
- Plots are much prettier than base plots (and imo, easier to manipulate)
- Part of the "tidyverse", authored by Wickham and collaborators, which also includes dplyr, purrr, tidyr, readr, and tibble.

Base plot vs. ggplot

Cars dataset: distance and speed, with a base scatterplot and a ggplot scatterplot:





I will often use base plots to make quick plots during exploratory data analysis (EDA) and ggplot for anything that will be presented/published to a non-data scientist audience.

Code for the last slide

Of course, the code that produces those two plots are different:

```
#base plot
plot(cars)

#ggplot
ggplot(cars, aes(speed, dist)) +
  geom_point()
```

And this is without any of the bells and whistles (i.e. themes, titles, fancier labels, etc.)

Lest you be intimidated...

- Off the top of my head, I can make very basic plots, usually with no additional features, in ggplot2.
- I have distinct memories of being scared of ggplots just 1 year ago. My graduate thesis contains either base plots or nothing.
- For everything else, I use the documentation website and StackOverflow. -I've also really benefitted from DataCamp, but it isn't free.
- ggplot2 Documentation can be found at: http://ggplot2.tidyverse.org/

The main reasons my ggplot2 code isn't working

- I've forgotten to close a parenthesis.

```
#this won't run
ggplot(rladies_chi_survey, aes(x=r_level) + geom_bar()
```

- I've put the + in the wrong place.

```
#this won't run either
ggplot(rladies_chi_survey, aes(x=r_level))
      + geom_bar()
```

The main reasons my ggplot2 code isn't working

- I've not exited a character string and tried to keep typing.

```
#this too will not run
ggp_indicator <- ggplot(data.frame(dataset_from_work),
                        aes(x=indic_var))
ggp_indicator + geom_histogram(fill="lightblue,
                               stat="count"")
```

- I really, really want geom_histogram() to be geom_hist(). (This one's inexplicable.)

Basics of ggplot2

```
ggplot(data = <DATA>) +
    <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

- Three things required by all ggplots:

1. dataset (DATA)
2. geom (GEOM_FUNCTION)
3. mappings (MAPPINGS)

- Variety of geoms depending on what you want to plot (geom_point, geom_line, geom_boxplot, etc.)
- Mappings are the variables you want to graph + other aesthetics (aes)
- Additionally, you can maniuplate aesthetics with custom themes and ggplot can calculate statistics for you.

Basics of ggplot2

I write my ggplot calls like this:

```
ggplot(data = <DATA>, aes(<GLOBAL MAPPINGS>)) +
    <GEOM_FUNCTION>(mapping = aes(<LOCAL MAPPINGS>))
```

- Global mappings will apply globally to every layer
- Local mappings will override or add on to any global mappings for that layer only.

Basics of ggplot2

- The plus signs should be at the END of a line, always!

```
ggplot(data = <DATA>, aes(<GLOBAL MAPPINGS>)) +
    <GEOM_FUNCTION>(mapping = aes(<LOCAL MAPPINGS>))
```

- You can save a ggplot call and then use it for multiple graphs.

```
#example from my job, on a generic Y1 outcome
ggp_Y1 <- ggplot(data.frame(work_dataset), aes(x=Y1))
#use the ggplot object + bar call:
ggp_Y1 + geom_bar(fill="lightblue")
#use the ggplot object + a geom_histogram call
    # + stat to create same graph:
ggp_Y1 + geom_histogram(fill="lightblue", stat="count")
```

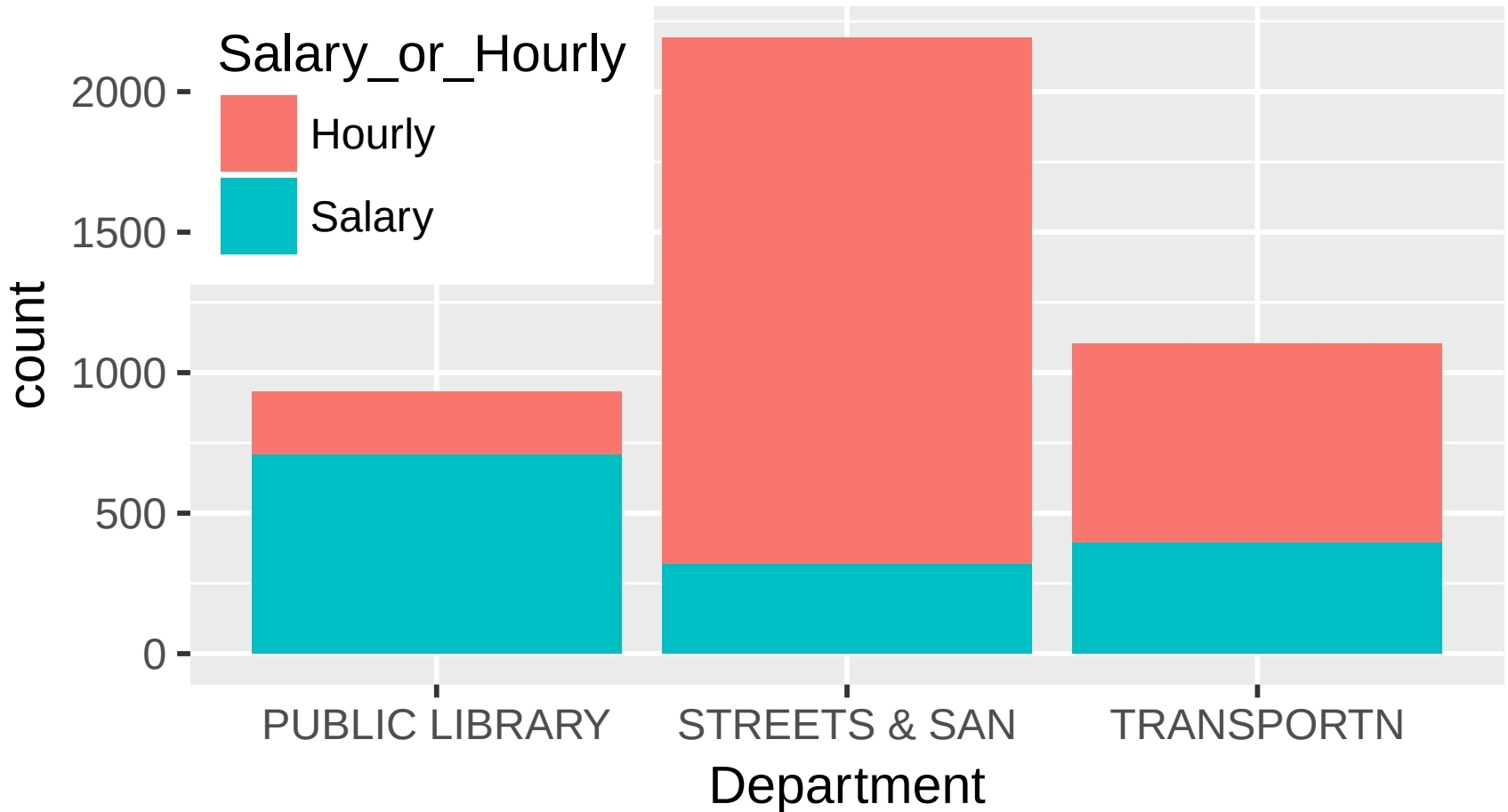
Basics of ggplot2

- Adding inside the aes() call means an aesthetic is mapped to the value of the variable in the data.
- Outside an aes call, the aesthetic is set to a specific value.
- Demonstrated on the next slide. . .

Aesthetics

- The fill is the Salary or hourly variable, and a legend appears to let me know which!

```
ggplot(chicago_3depts,
       aes(x=Department, fill=Salary_or_Hourly)) +
  geom_bar() + theme(legend.justification=c(0,1), legend.position=c(0,1
```



Aesthetics

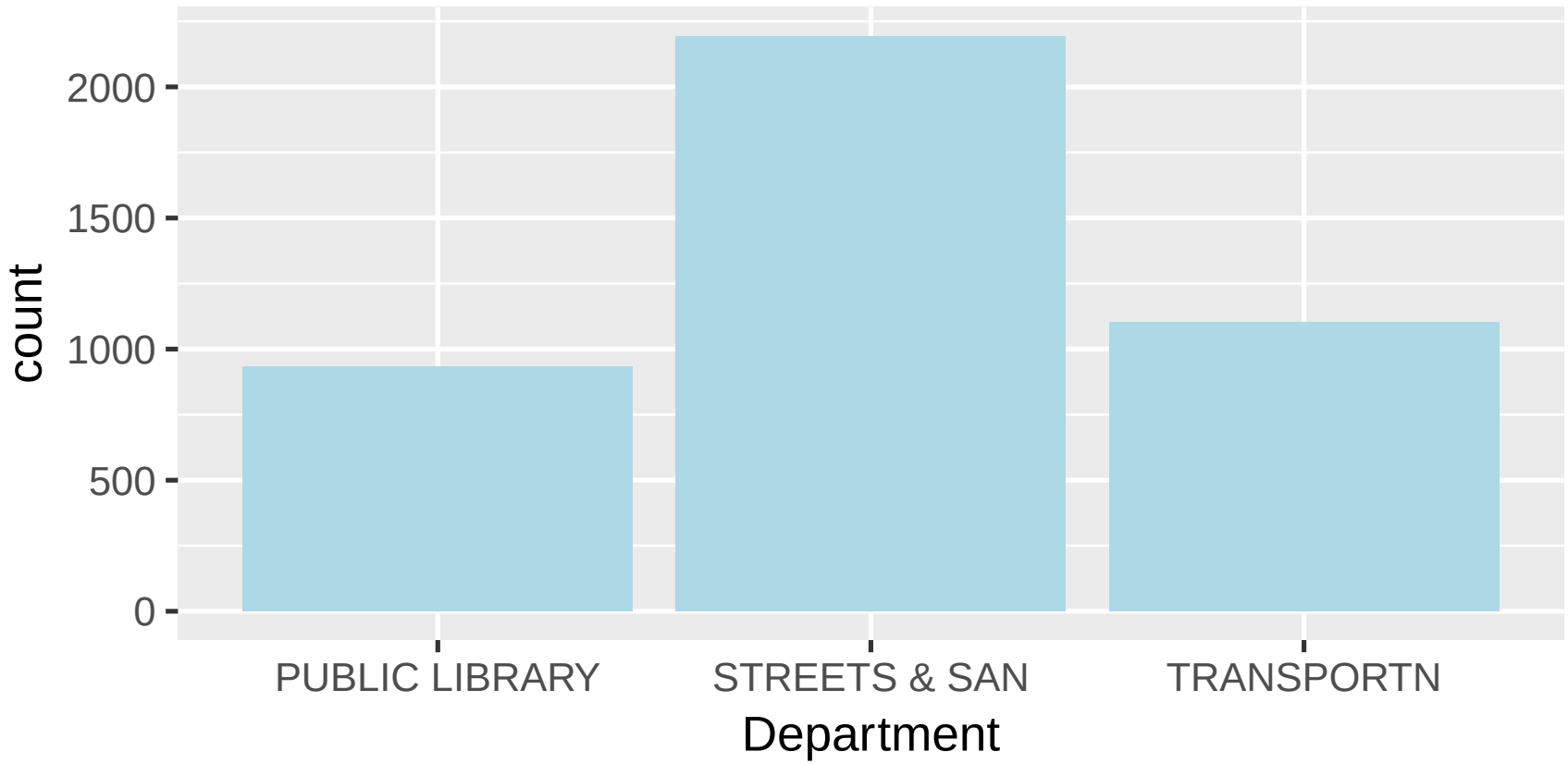
- With this code, it now thinks fill is a thing called "lightblue". I wanted the bars to be colored light blue but have it in the wrong place.

```
ggplot(chicago_3depts, aes(x=Department, fill="lightblue")) +
  geom_bar()
```

Aesthetics

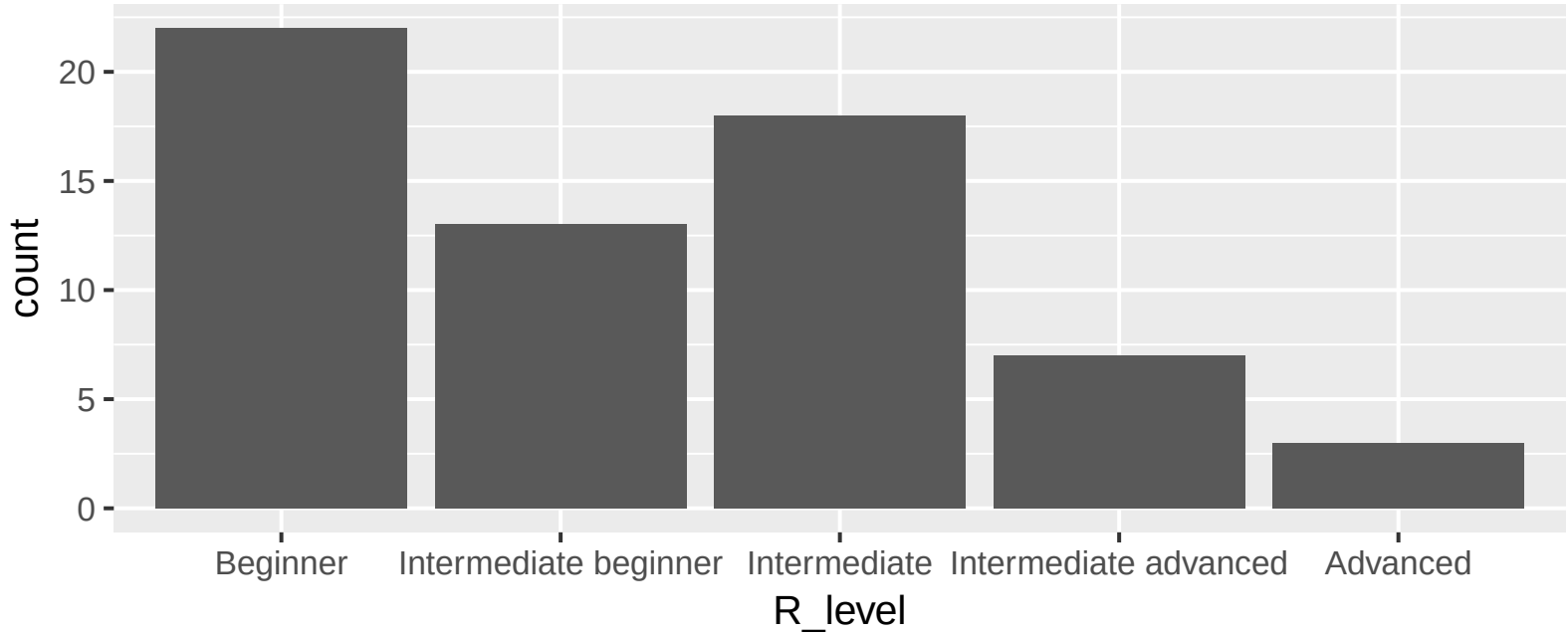
- To get lightblue bars, you have to put the fill argument in geom_bar() so it's local. Voila!

```
ggplot(chicago_3depts, aes(x=Department )) +
  geom_bar(fill="lightblue")
```



One example from the August RLadies Chicago survey

- geom_bar takes only one mapping (x=Variable) and returns count of the levels of that variable by default.



- Aaaand now I'm going to abandon this data–it doesn't work for the purposes of showing examples across this talk very well.

Dataset for the rest of the talk

- Chicago salary data for all employees of the City of Chicago as of May 4, 2017
- Available on Kaggle as an open dataset (and also through the City's open data portal, but I got it off Kaggle)
- Kaggle link: https://www.kaggle.com/chicago/chicago-citywide-payroll-data/data

```
names(chicago_salary_data)
```

```
## [1] "Name"                "Job_Titles"          "Department"
## [4] "Full_or_Part-Time" "Salary_or_Hourly"  "Typical_Hours"
## [7] "Annual_Salary"     "Hourly_Rate"
```

Some data cleaning

- To make it easier on myself (and to cut down on the sheer number of categories in "Department"), I created a dataset with only 3 departments: Public Library, Streets & Sanitation, and Transportation employees
- This gave me a good mix of full- and part-time employees.
- I called this dataset chicago_3depts, and probably could have named it better.

```
library(dplyr)
chicago_3depts <- chicago_salary_data %>%
    filter(chicago_salary_data$Department=="PUBLIC LIBRARY" |
             chicago_salary_data$Department=="STREETS & SAN" |
             chicago_salary_data$Department=="TRANSPORTN" )
```

Data cleaning, con't

- Because of this combination of salaried and hourly employees, where salaried employees have a value in the Annual Compensation variable and hourly do not, I wanted to create a total_compensation variable.

```
##
## Hourly Salary
##   2804   1425
```

Data cleaning, con't

The following code does just that:

```
#i assumed a 52 week year, which may be unrealistic
chicago_3depts$Total_compensation <-
  ifelse(is.na(chicago_3depts$Annual_Salary),
         chicago_3depts$Typical_Hours*chicago_3depts$Hourly_Rate*52,
         chicago_3depts$Annual_Salary)
```

Resulting in:

```
## # A tibble: 6 x 1
##   Total_compensation
##                <dbl>
## 1            74048.0
## 2            43201.6
## 3            74048.0
## 4            26104.0
## 5            75316.8
## 6            74048.0
```

Data cleaning, con't

I also created a dataset each, separating Salaried and Hourly employees:

```
chicago_3depts_Salary <- chicago_3depts %>%
      filter(chicago_3depts$Salary_or_Hourly == "Salary")

chicago_3depts_Hourly <- chicago_3depts %>%
      filter(chicago_3depts$Salary_or_Hourly == "Hourly")
```

Exploratory Data Analysis

Here's where I could start making tables and calling stats functions to explore the data:

```
table(chicago_3depts$Department)
```

```
##
## PUBLIC LIBRARY  STREETS & SAN     TRANSPORTN
##            932           2194           1103
```

```
mean(chicago_3depts$Total_compensation)
```

```
## [1] 71568.63
```

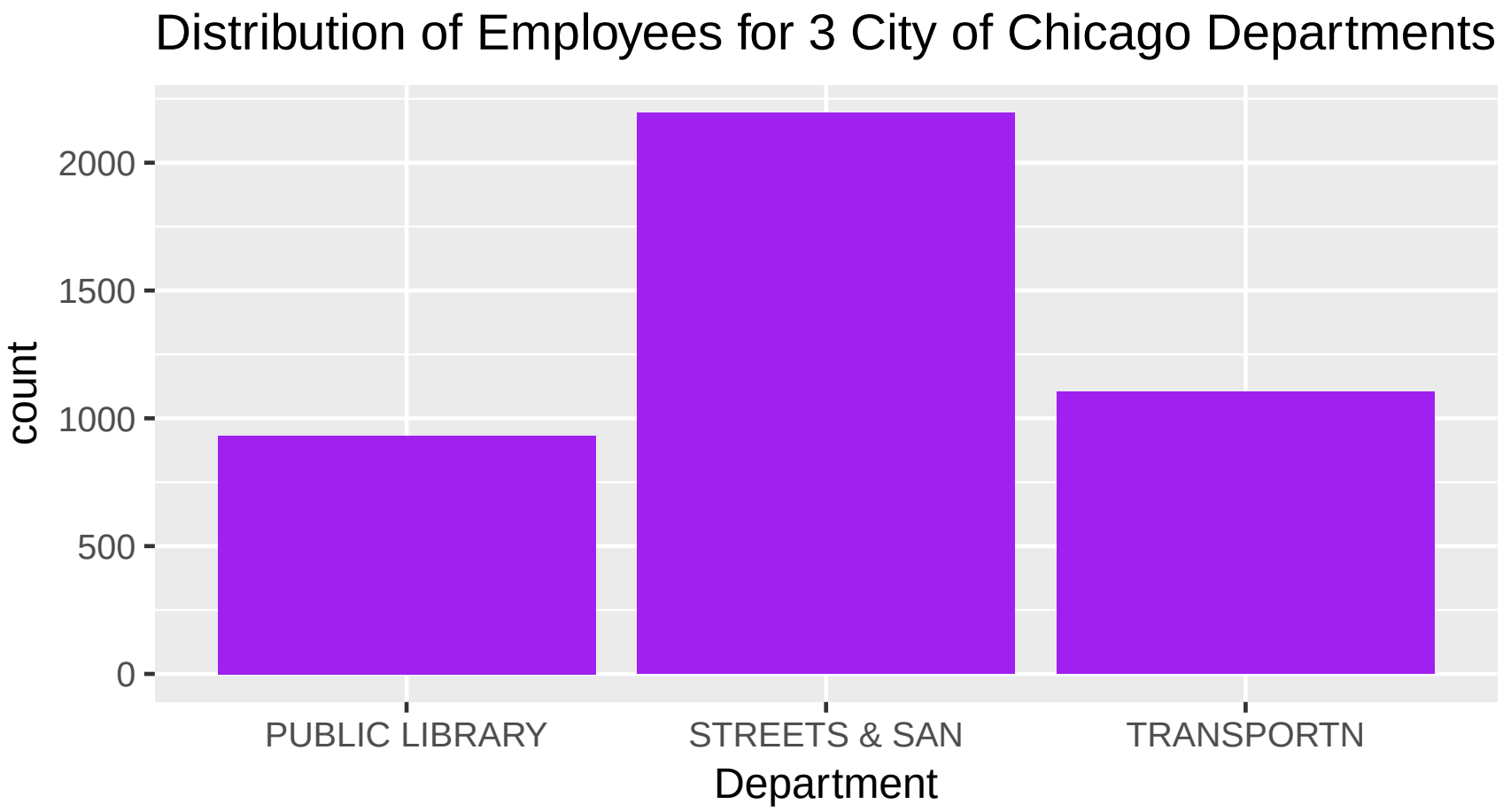
Exploratory Data Analysis

Or even calls using dplyr:

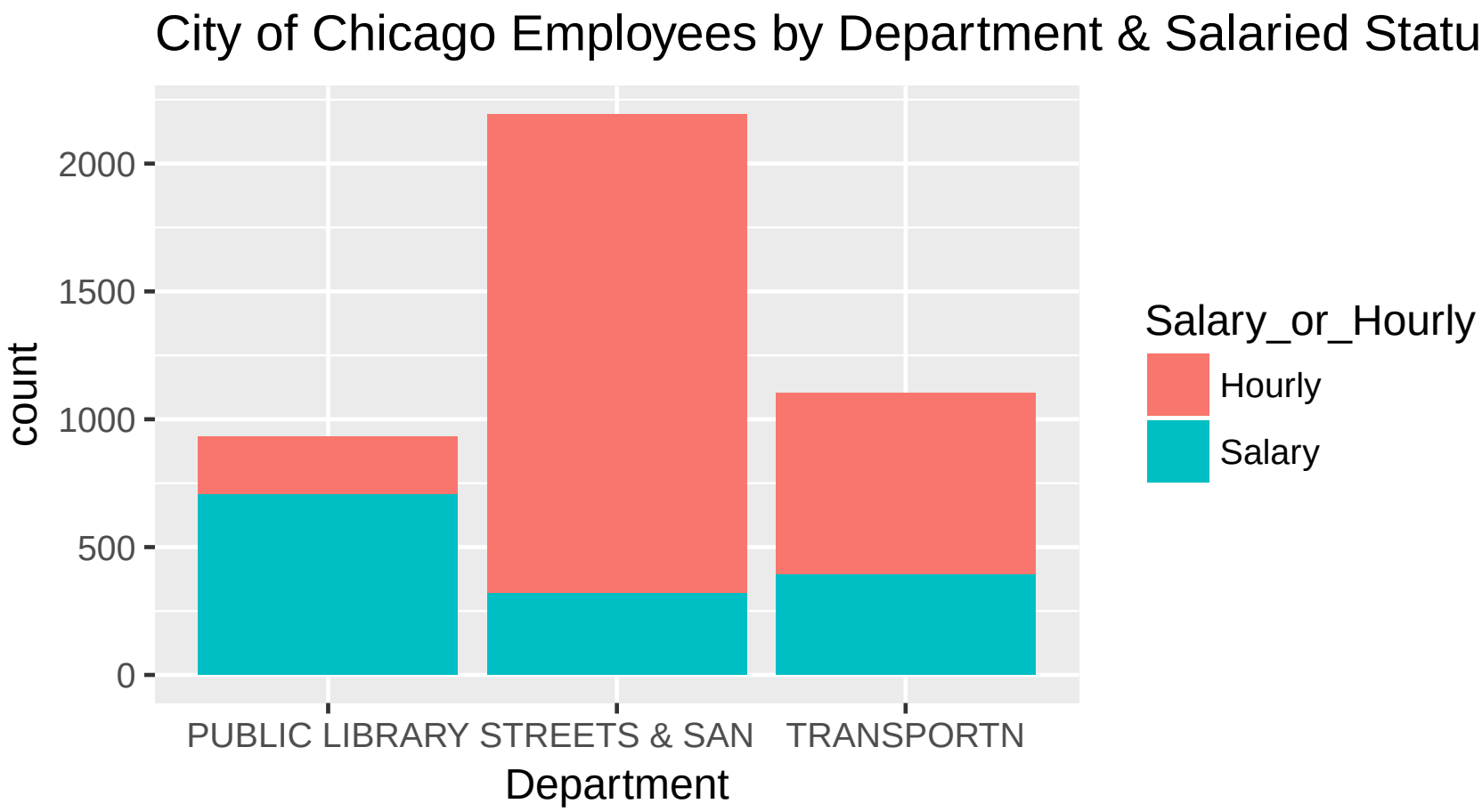
```
chicago_3depts %>%
  filter(!is.na(chicago_3depts$Hourly_Rate)) %>%
  summarise(mean=mean(Hourly_Rate),
            median=median(Hourly_Rate))
```

```
## # A tibble: 1 x 2
##       mean median
##      <dbl>  <dbl>
## 1 34.54159  36.13
```

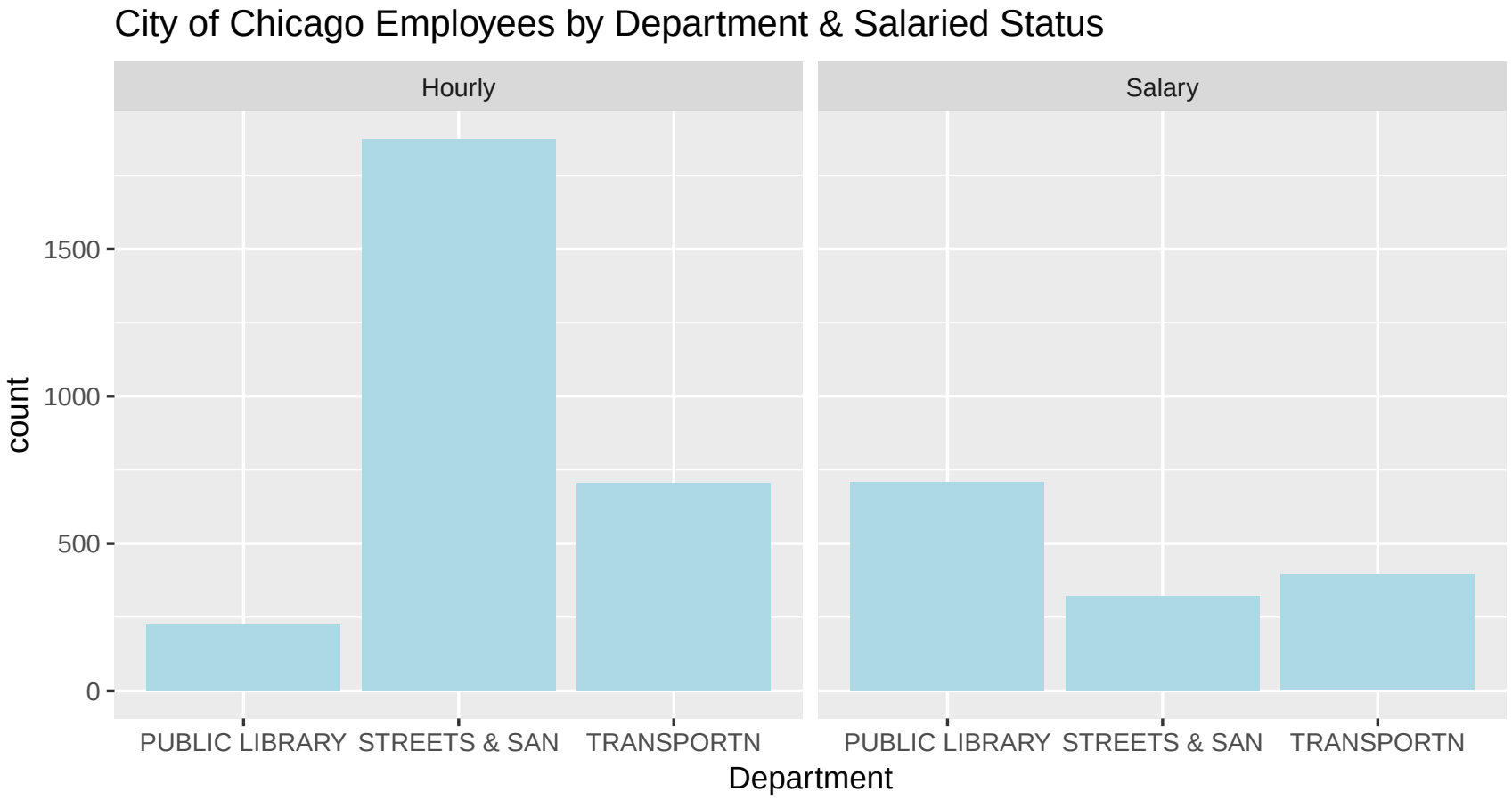
Or I can visualise it!



Or I can visualise it!



Or I can visualise it!



Bar Charts vs. Histograms

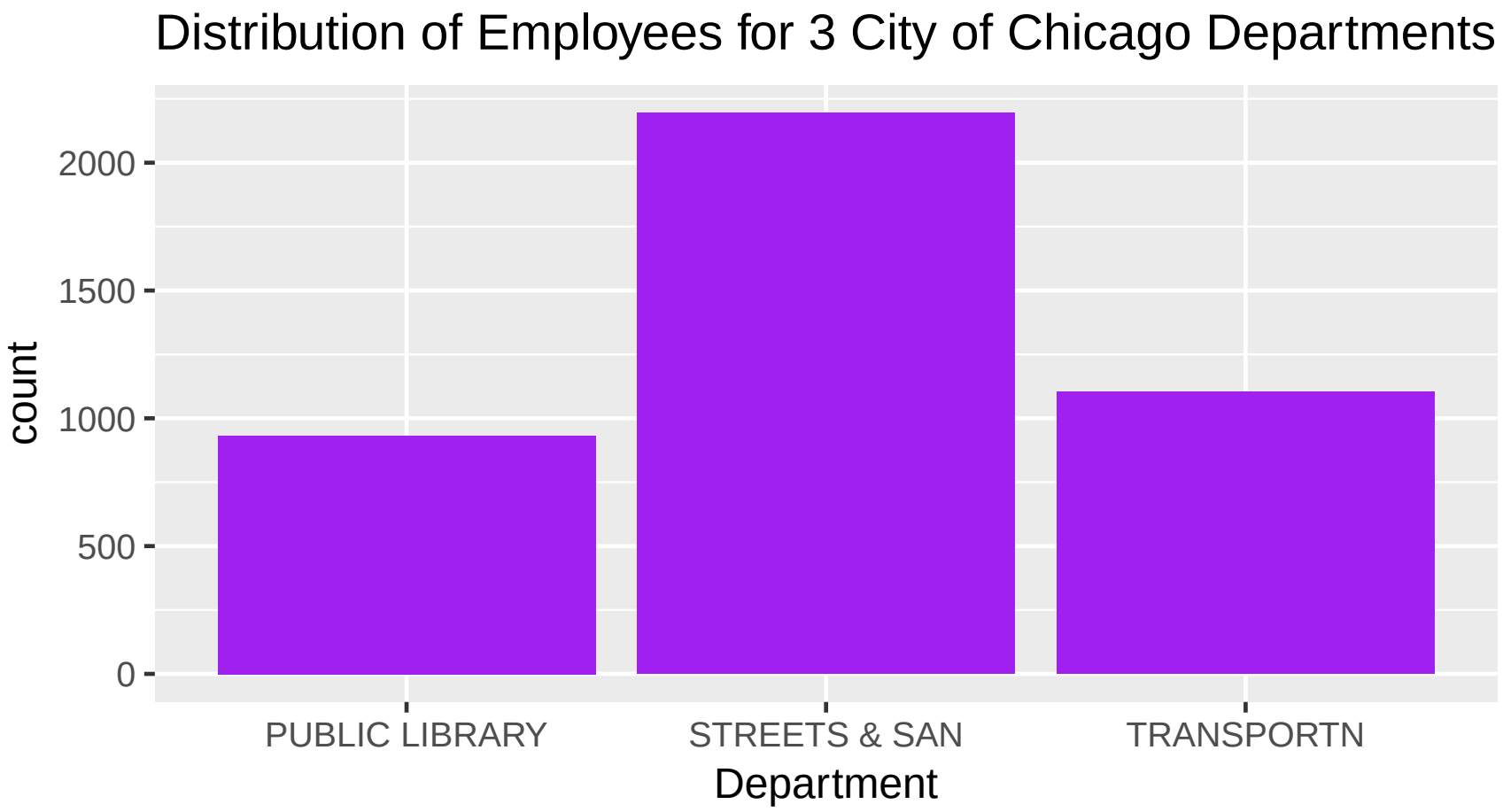
- Bar charts, executed in with geom_bar, are used to display categorical data.
- Histograms, executed with geom_histogram, are used to display interval (usually numeric) data.
- We can use both types with our Chicago Salary data (the last slide was all bar charts.)
- You can create bar charts with a stat call in geom_histogram, to be demonstrated.

Bar Charts - Plot 1

Let's walk through the code of the bar charts I just showed:
- only aesthetic is x, the variable I want to display
- overrwrote the color fill in geom_bar() to be purple
- added a title with ggtitle (I've broken the title into two lines to display easier in the LaTeX slides. If you do this in the code, the title will also be 2 lines.)

```
#plot 1
ggplot(chicago_3depts, aes(x=Department)) +
  geom_bar(fill="purple") +
  ggtitle("Distribution of Employees
          for 3 City of Chicago Departments")
```

Plot 1

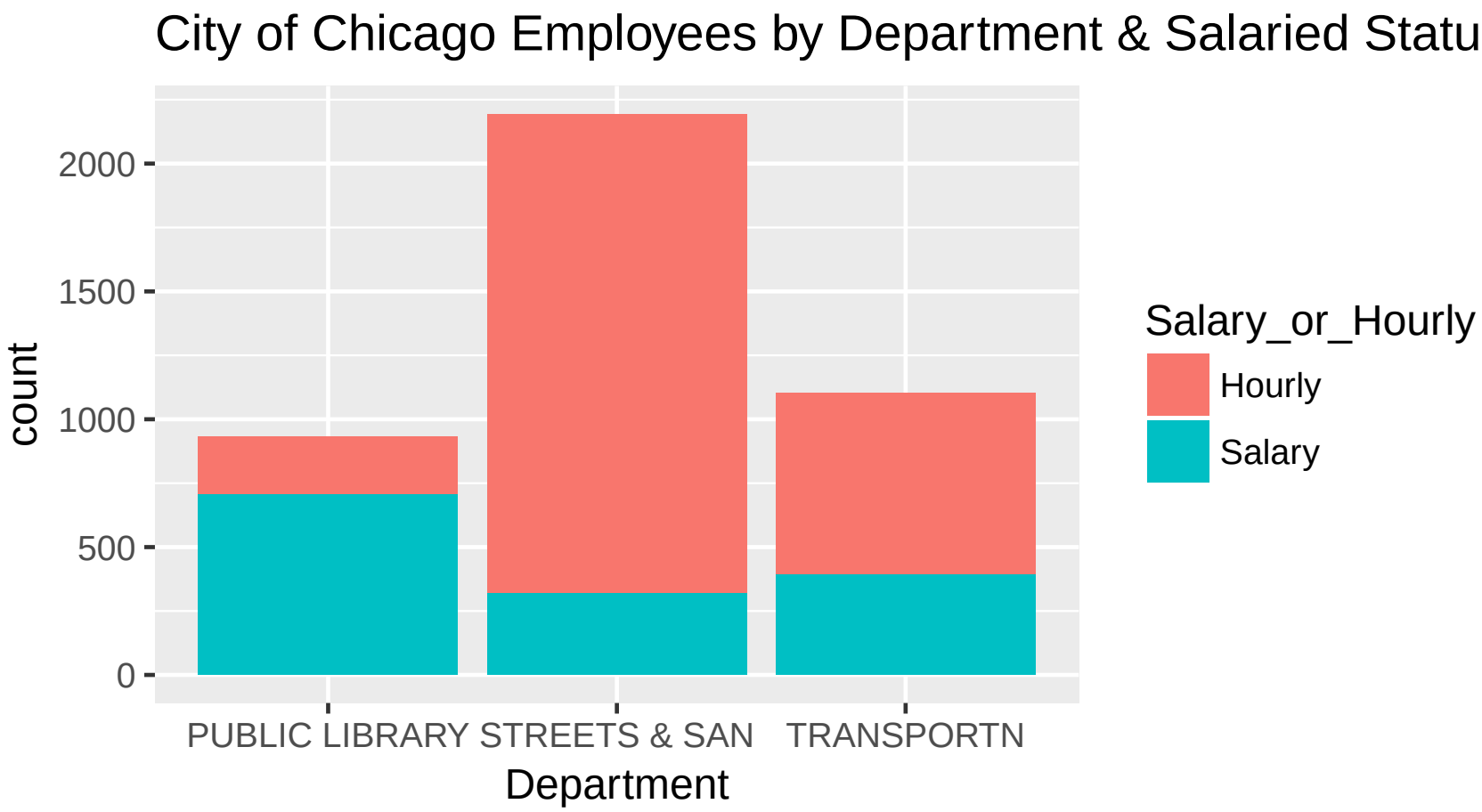

Distribution of Employees for 3 City of Chicago Departments
count
2000
1500
1000
500
0
PUBLIC LIBRARY
STREETS & SAN
TRANSPORTN
Department

Bar Charts - Plot 2

- first aesthetic is x, the variable I want to display
- then, on ggplot line, I added fill = Salary_or_Hourly, so I can fill the bars based on counts of which employees are salary and which are hourly
- added a title with ggtitle

```
#plot 2
ggplot(chicago_3depts, aes(x=Department, fill=Salary_or_Hourly)) +
  geom_bar() +
  ggtitle("City of Chicago Employees by
          Department and Salaried Status")
```

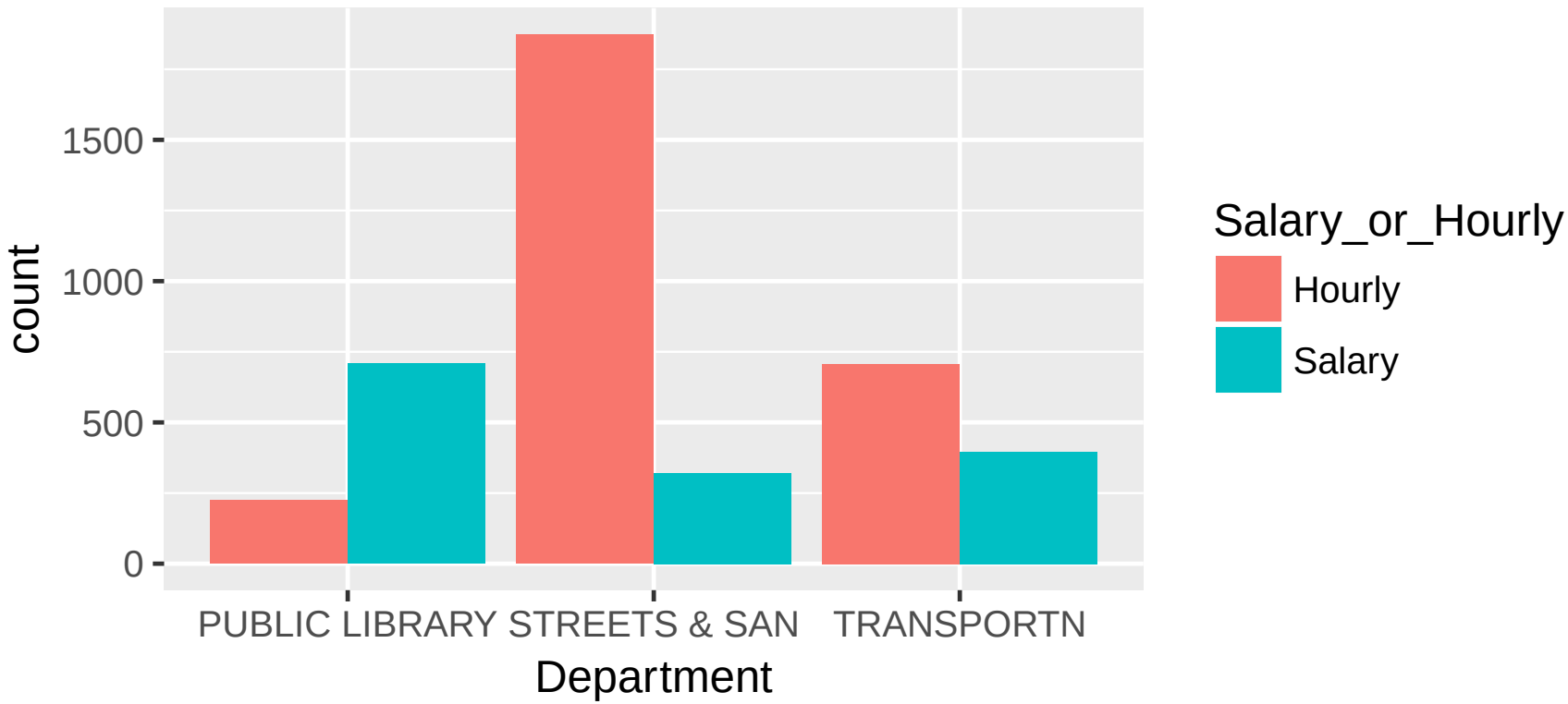
Plot 2


City of Chicago Employees by Department & Salaried Status
count
2000
1500
1000
500
0
PUBLIC LIBRARY
STREETS & SAN
TRANSPORTN
Department
Salary_or_Hourly
Hourly
Salary

Bar Charts - Dodge

- by adding position = "dodge" to geom_bar, the Salary_or_Hourly bars will be next to one another instead of filling the original bars by their counts

```
#plot 2
ggplot(chicago_3depts, aes(x=Department, fill=Salary_or_Hourly)) +
  geom_bar(position="dodge")
```



Bar Charts - Fill

- by adding position = "fill" to geom_bar, we get the percentage of each department that are hourly or salaried

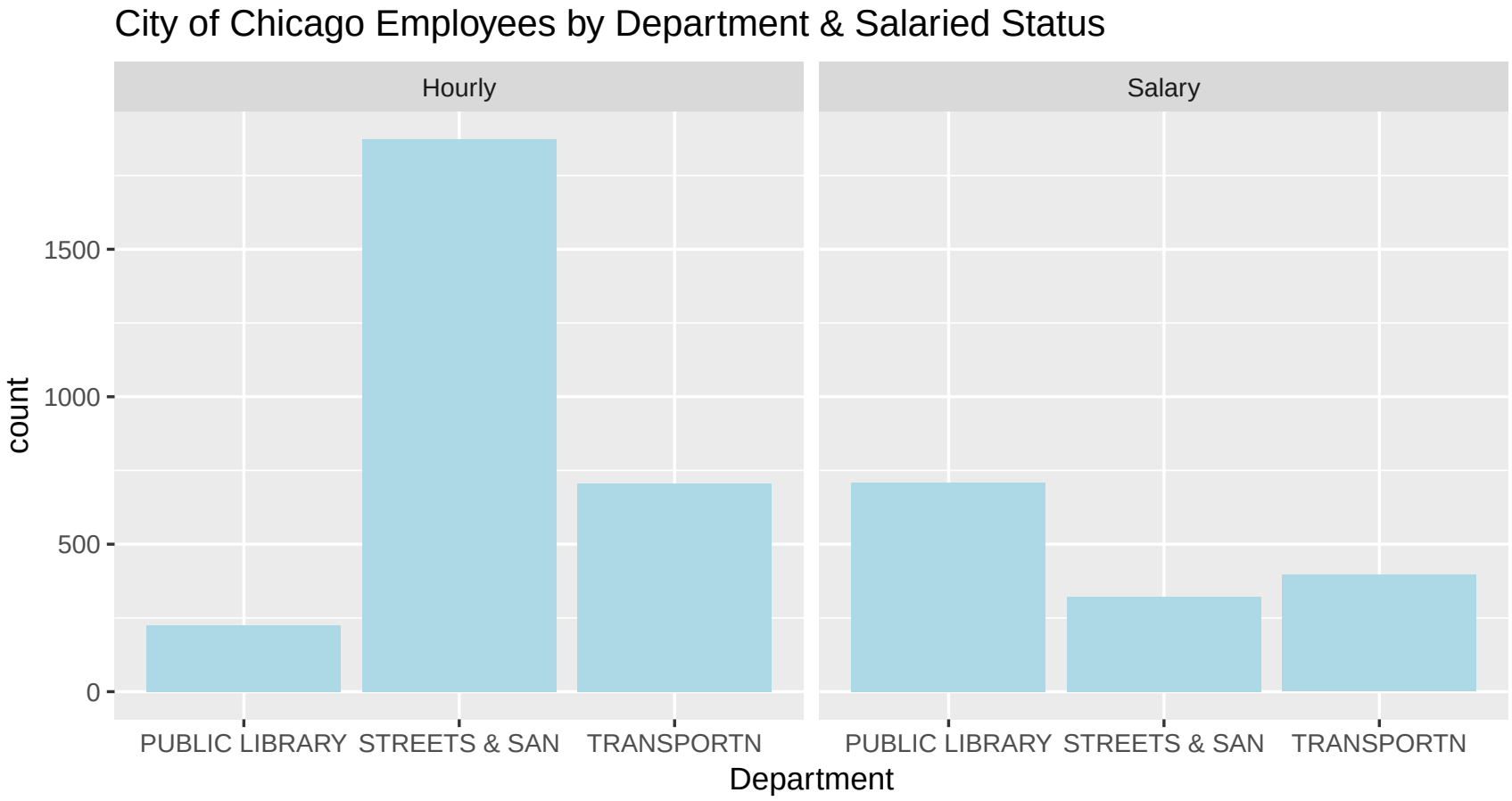
```
#plot 2
ggplot(chicago_3depts, aes(x=Department, fill=Salary_or_Hourly)) +
  geom_bar(position="fill")
```

Bar Charts - Plot 3

- The x aesthetic is still Department
- I changed the fill to be lightblue
- Added a title with ggtitle
- But now instead of making the global aesthetic fill = Salary_or_Hourly, I used facet wrapping to divide the graphs. I get counts of how many employees in each deparment, where one graph is Salaried and one Hourly employees.

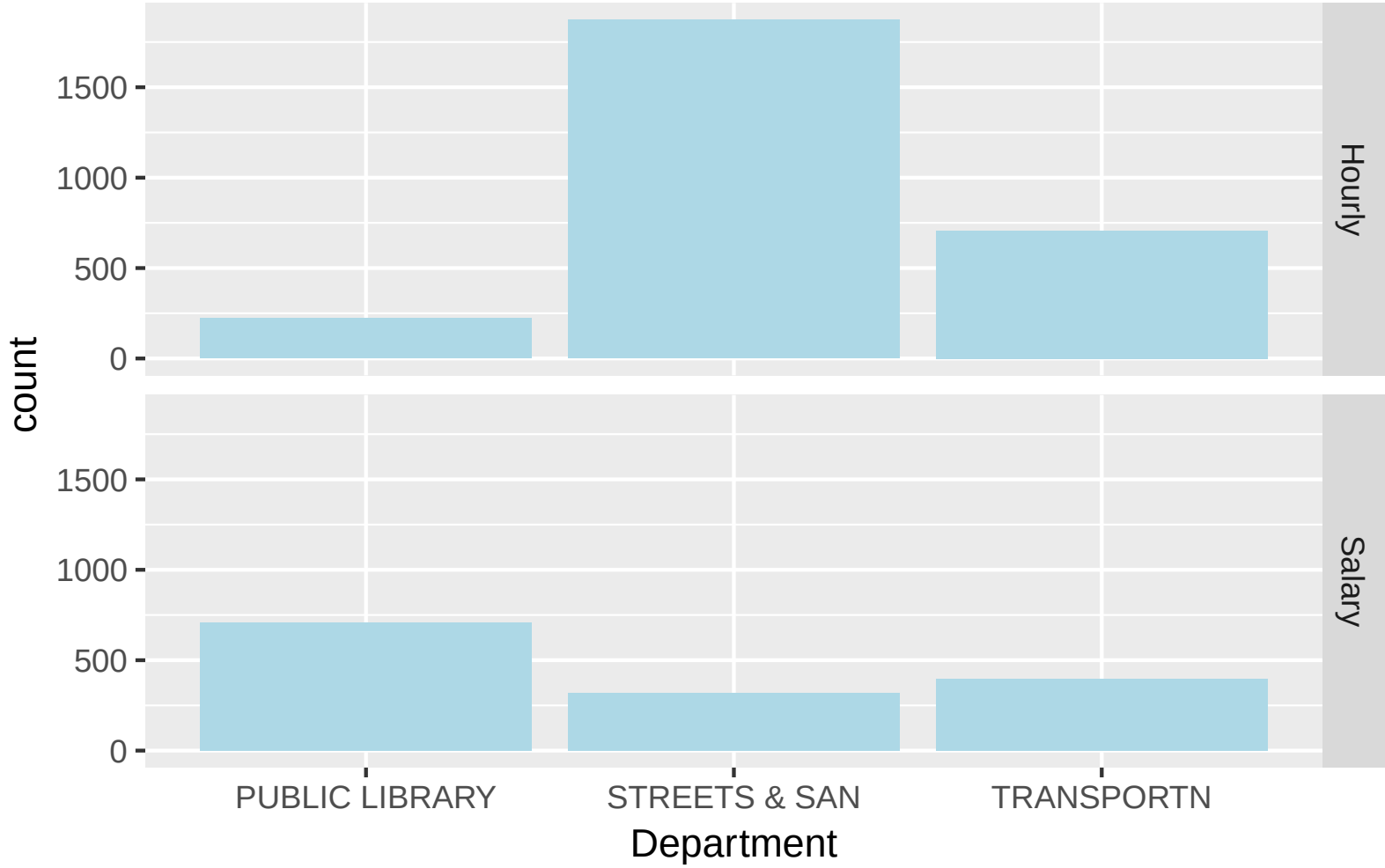
```
#plot 3
ggplot(chicago_3depts, aes(x=Department)) +
  geom_bar(fill="lightblue") +
  ggtitle("City of Chicago Employees by
          Department and Salaried Status") +
  facet_wrap(~Salary_or_Hourly)
```

Plot 3


City of Chicago Employees by Department & Salaried Status
Hourly
Salary
count
1500
1000
500
0
PUBLIC LIBRARY
STREETS & SAN
TRANSPORTN
PUBLIC LIBRARY
STREETS & SAN
TRANSPORTN
Department

Bar Charts - Facet Wrap/Grid

- I can also use facet gridding, which allows me to facet on either the (row ~ column)



YOUR TURN

- Let's use `mtcars`, a classic R dataset located in the datasets package

```
data(mtcars)
```

- Some info:

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Helpful Tips

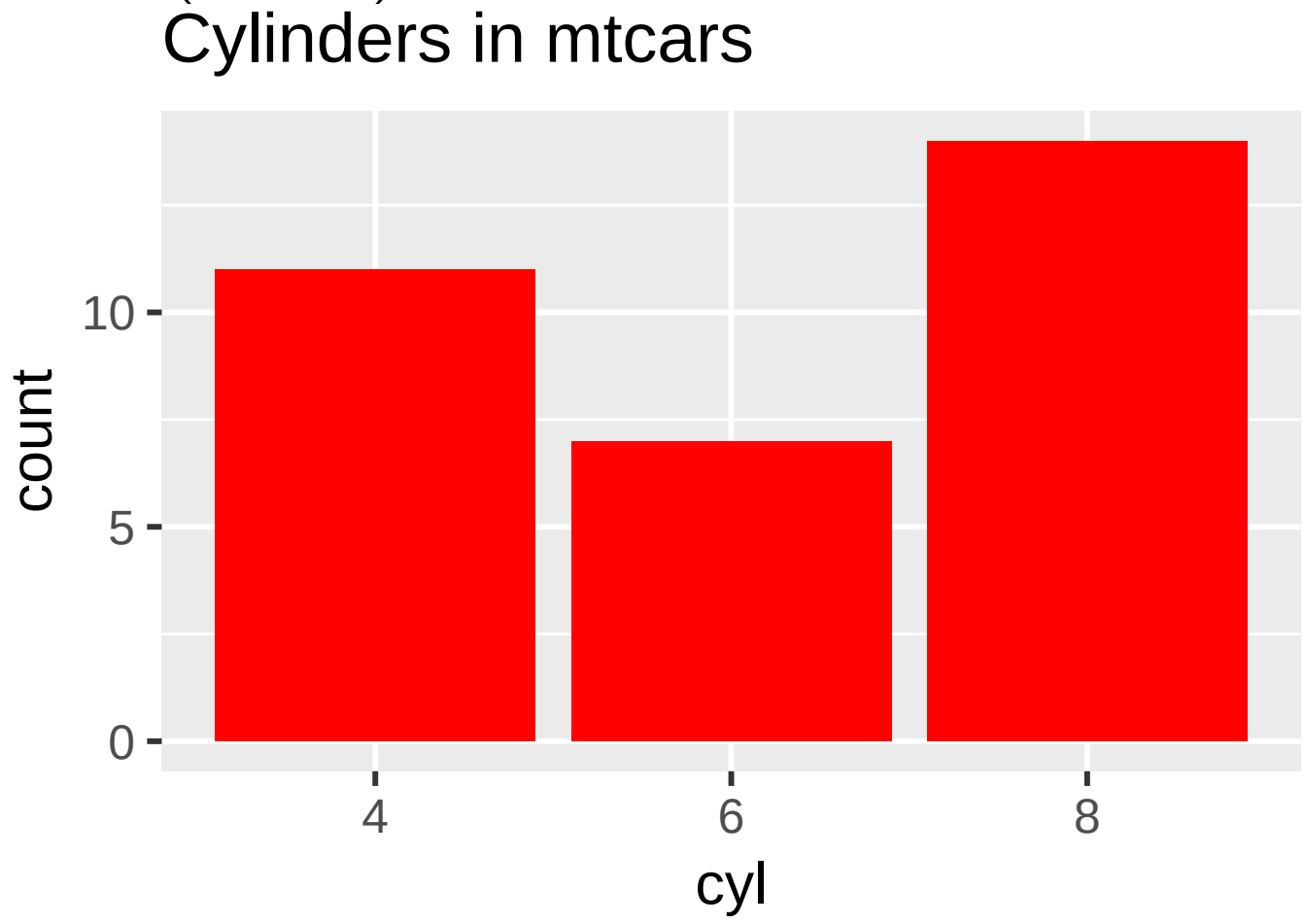
- Turn these three variables into factors first:

```
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$am <- as.factor(mtcars$am)
mtcars$gear <- as.factor(mtcars$gear)
```

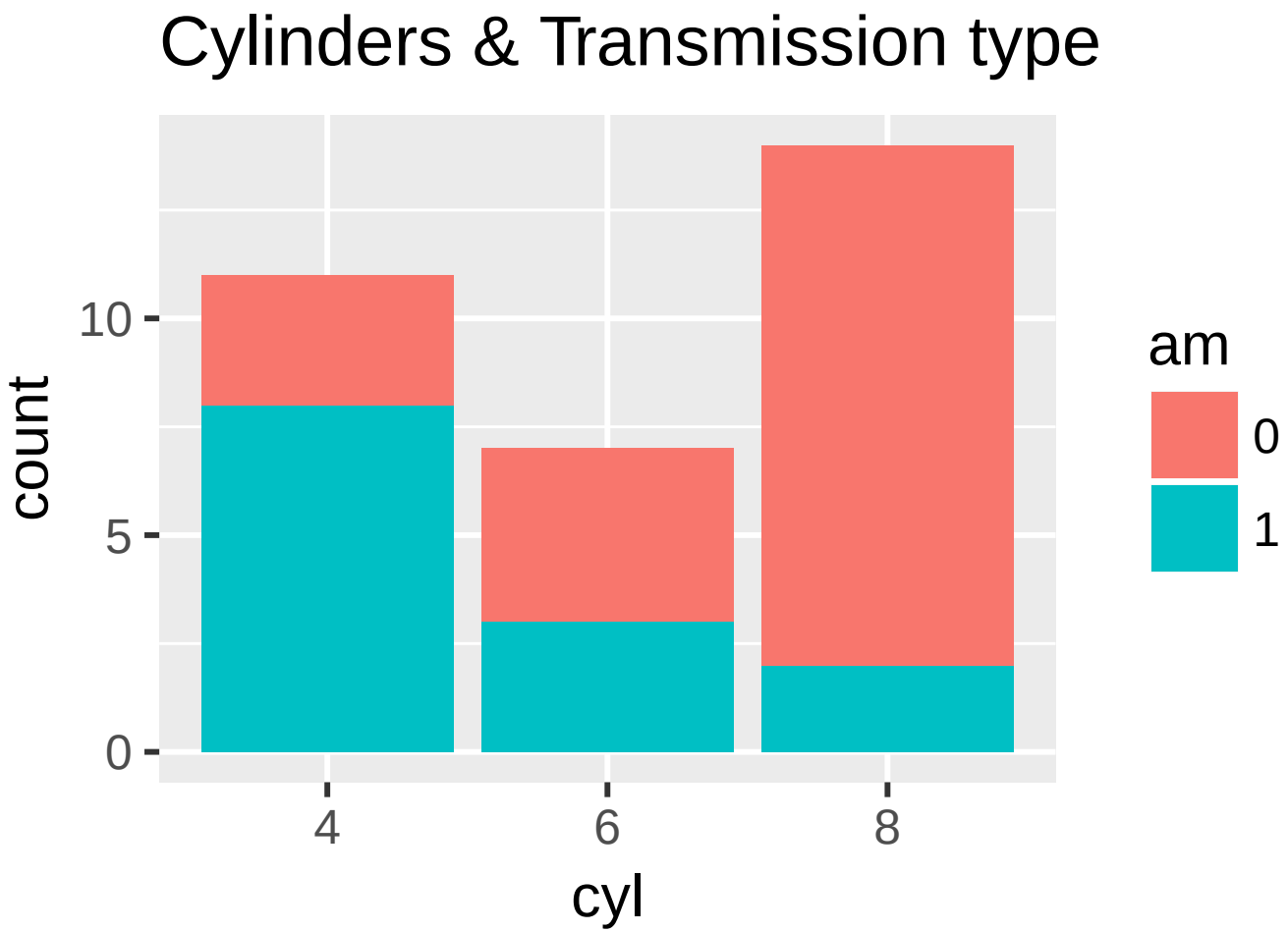
- This will help throughout
- Alternatively, you can just use factor(cyl) when you call any of them in your ggplot code (this will force the variable to be a factor in the plot.)

Recreate these bar charts!!!

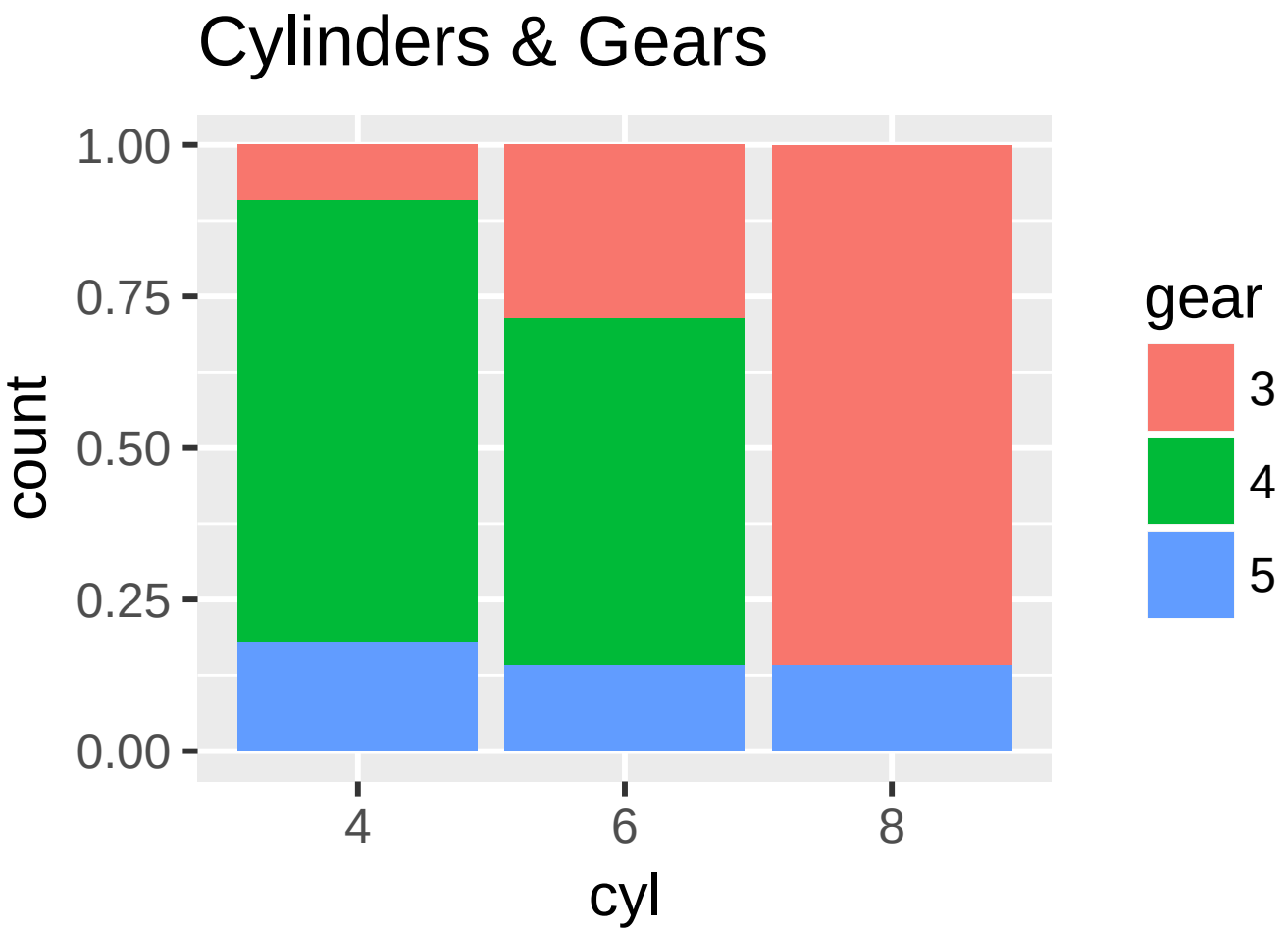
- Individually or in groups, try to recreate the following bar charts (3 slides):



Recreate these bar charts!!!



Recreate these bar charts!!!

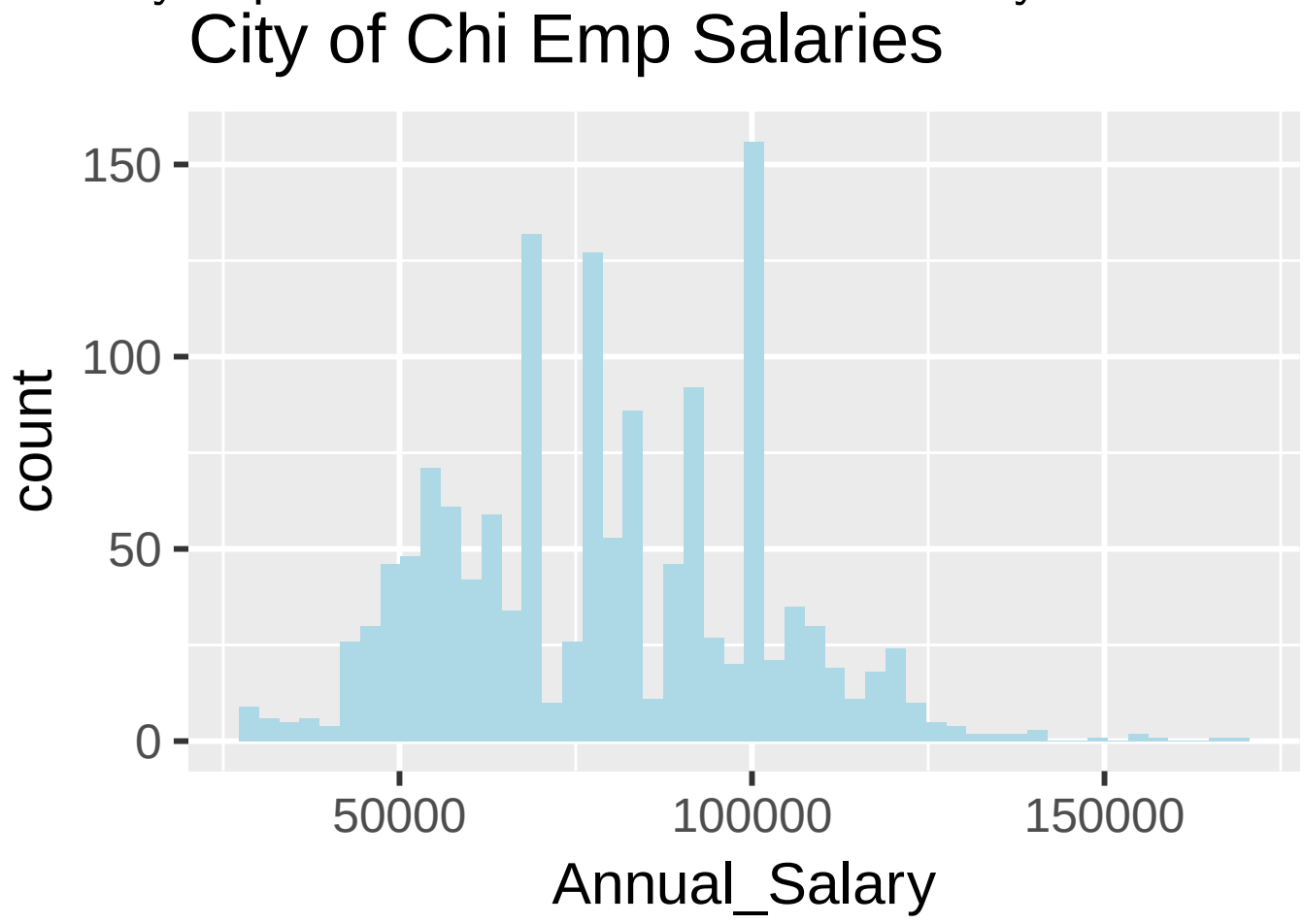


Solutions

```
ggplot(mtcars, aes(x=cyl)) +
  geom_bar(fill="red") +
  ggtitle("Cylinders in mtcars")

ggplot(mtcars, aes(x=cyl, fill=am)) +
  geom_bar() +
  ggtitle("Cylinders & Transmission type")

ggplot(mtcars, aes(x=cyl, fill=gear)) +
  geom_bar(position = "fill") +
  ggtitle("Cylinders & Gears")
```

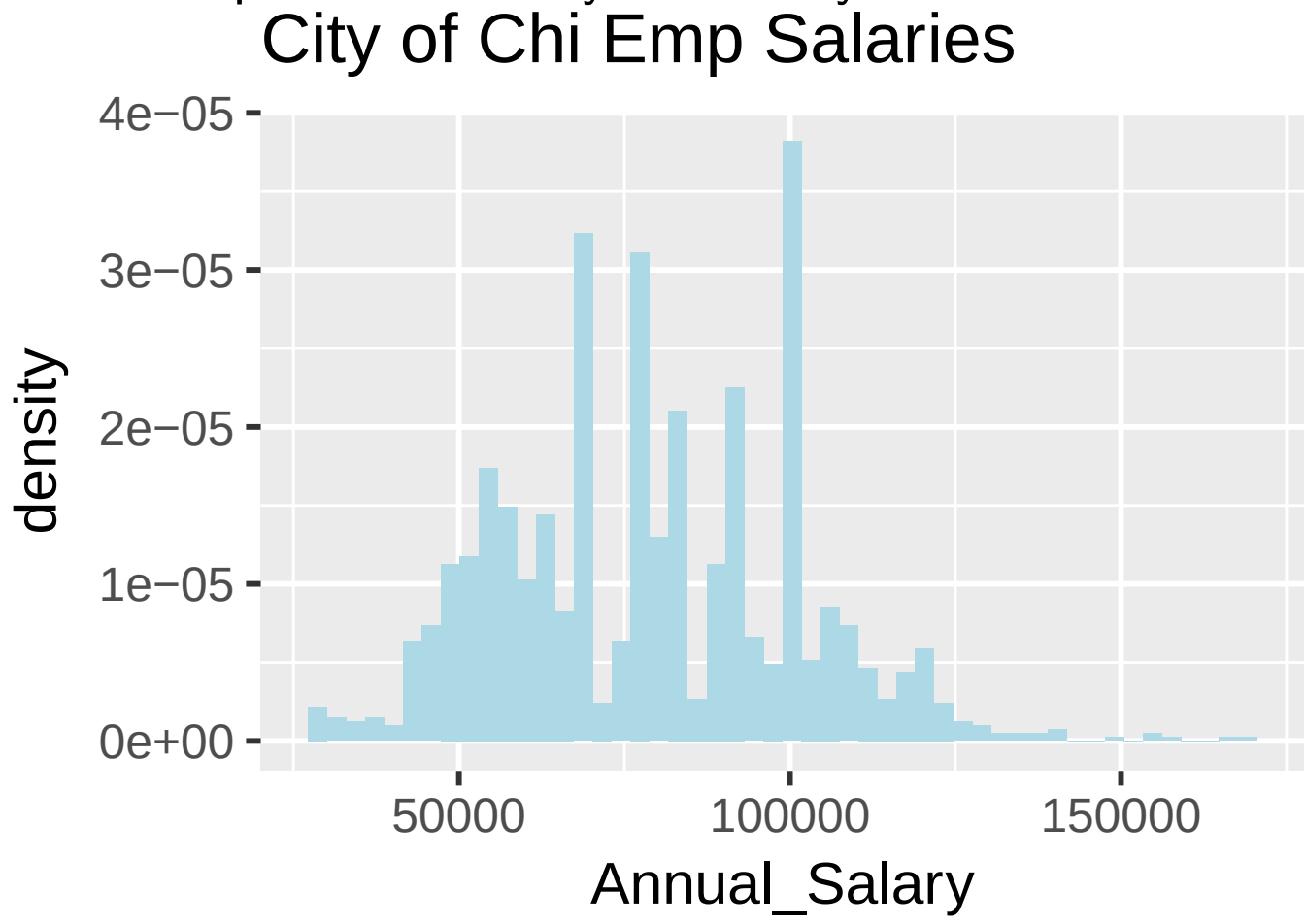
Histograms

- Notice we still get 'count' on the y-axis like a bar chart, but it's a histogram.
- The default binwidth is 30, and you often need to change this to fully explore the actual distribution of your data.



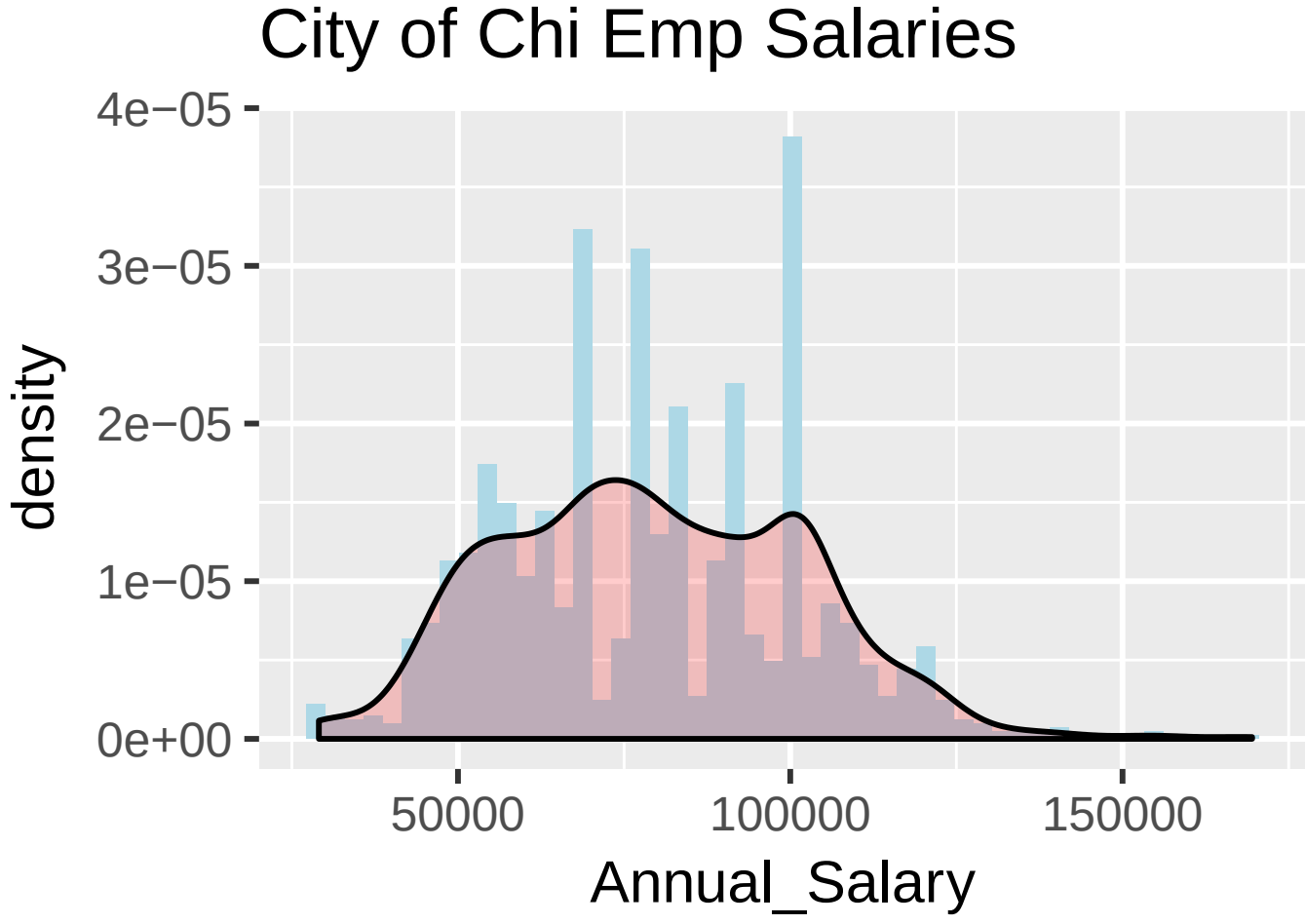
Histograms

- We could also plot the density or density + an overlaid density plot.



Histograms

- Or density + an overlaid density plot.



Histograms

```
#1st plot -- count histogram
ggplot(chicago_3depts_Salary, aes(x=Annual_Salary)) +
  geom_histogram(fill="lightblue", bins=50) +
  ggtitle("A More Meaningful Title")

#2nd plot -- density histogram
ggplot(chicago_3depts_Salary, aes(x=Annual_Salary)) +
  geom_histogram(fill="lightblue", aes(y=..density..), bins=50) +
  ggtitle("Than the Ones I Used")

#3rd plot -- density histogram + overlaid density plot
ggplot(chicago_3depts_Salary, aes(x=Annual_Salary)) +
  geom_histogram(fill="lightblue", aes(y=..density..), bins=50) +
  geom_density(alpha=0.2, fill="red") +
  ggtitle("To Save space with beemer")
```

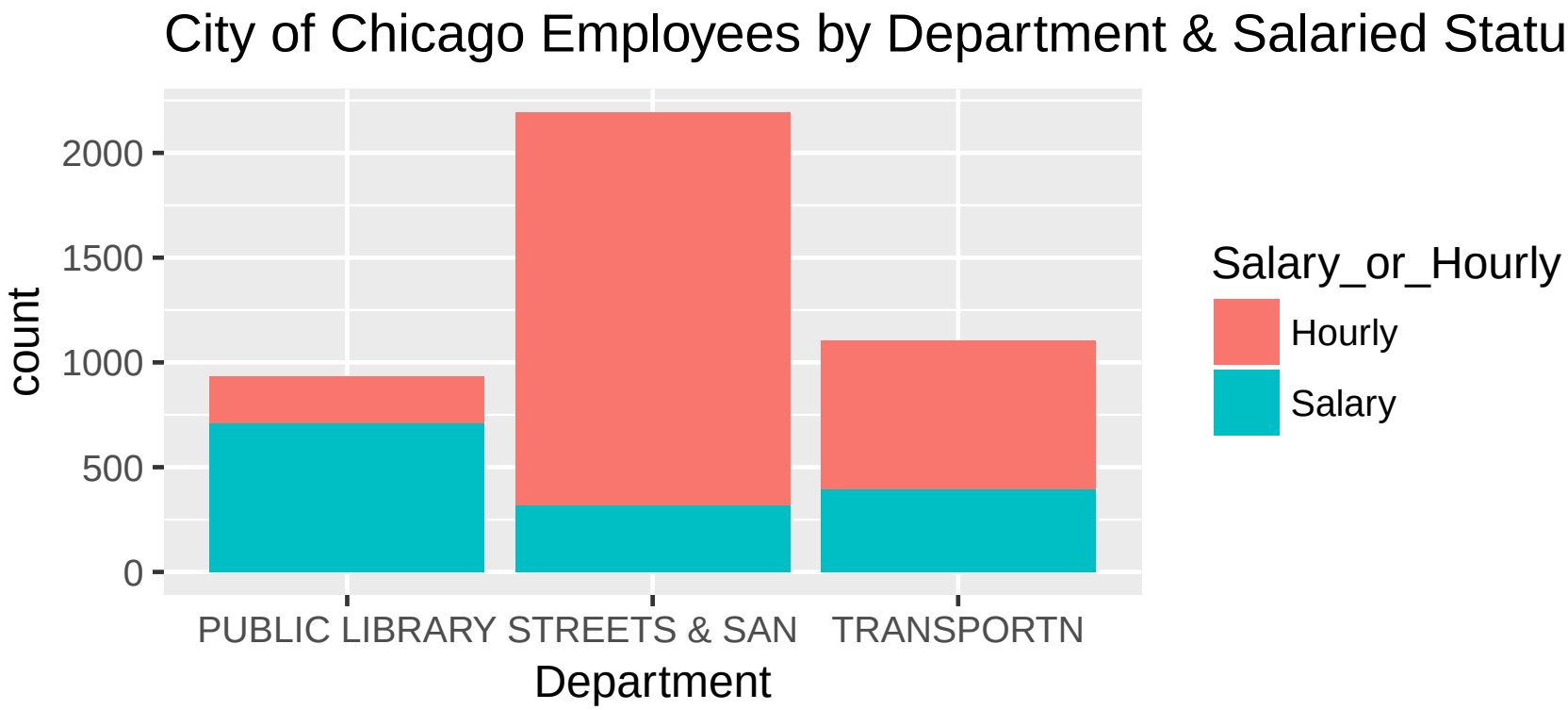
Histograms + stat call

- We could get some of the same bar charts as above with geom_histogram also
- To do so, we just have to add a stat= call inside geom_histogram:
- If needed, position arguments ("dodge", "fill") will also still work with geom_histogram.
- It throws a warning, but does recreate the bar chart successfully.
- Bar chart w/geom_histogram() on next slide.

Histograms + stat call

```
ggplot(chicago_3depts, aes(x=Department, fill=Salary_or_Hourly)) +
  geom_histogram(stat="count") +
  ggtitle("City of Chicago Employees by Department & Salaried Status")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



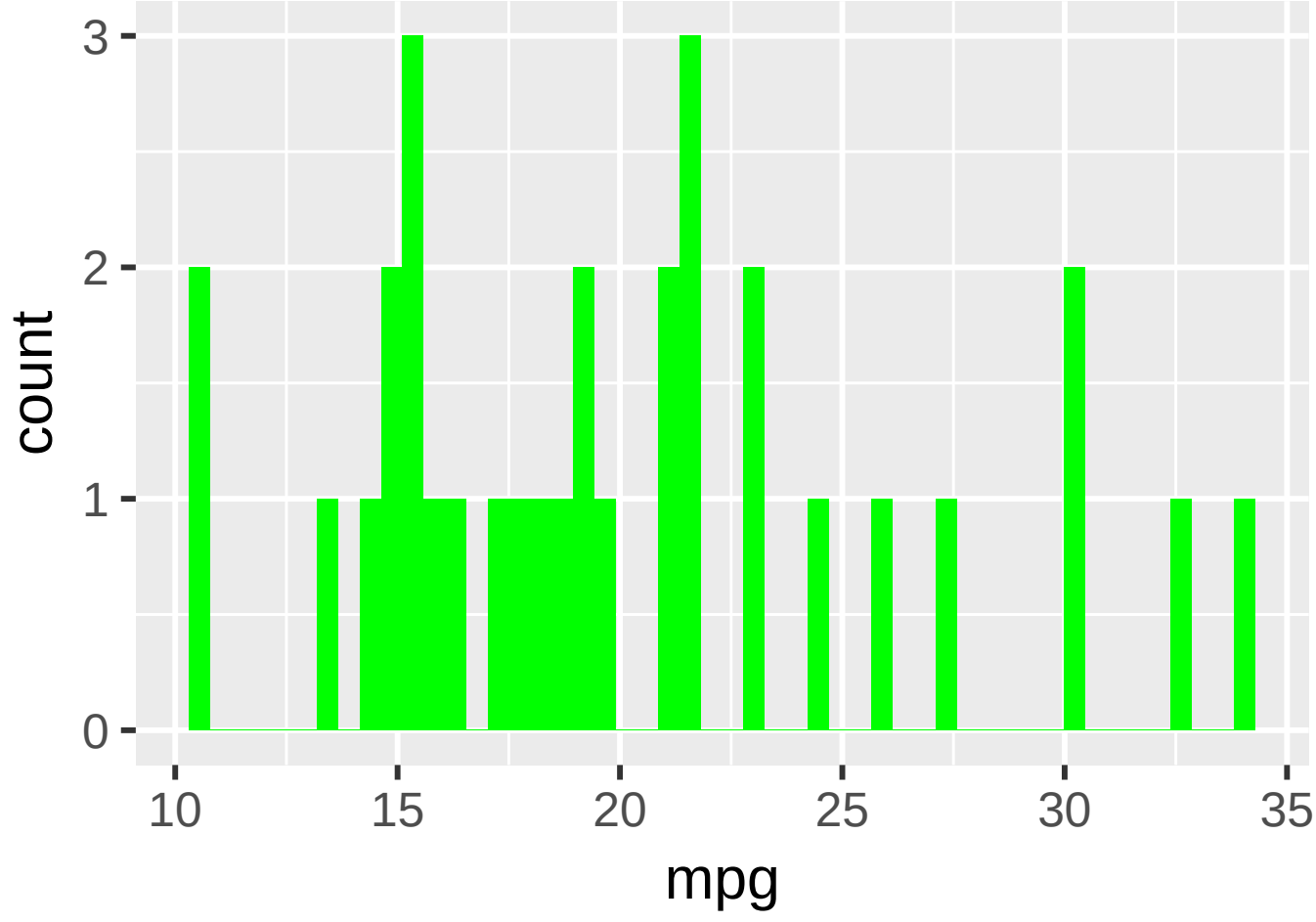
YOUR TURN

- `mtcars` again!

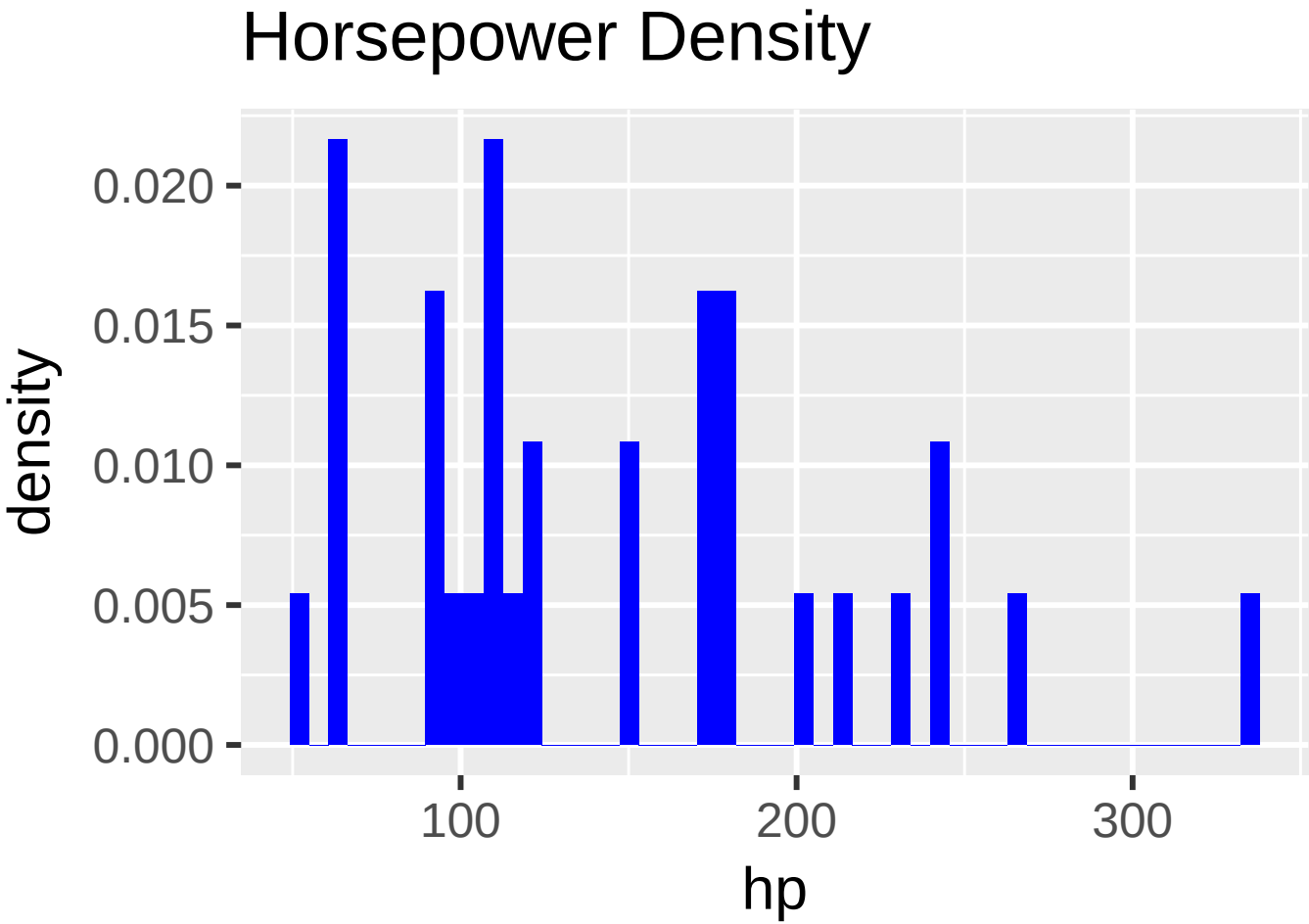
```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
##  $ gear: Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Recreate these histograms!!!

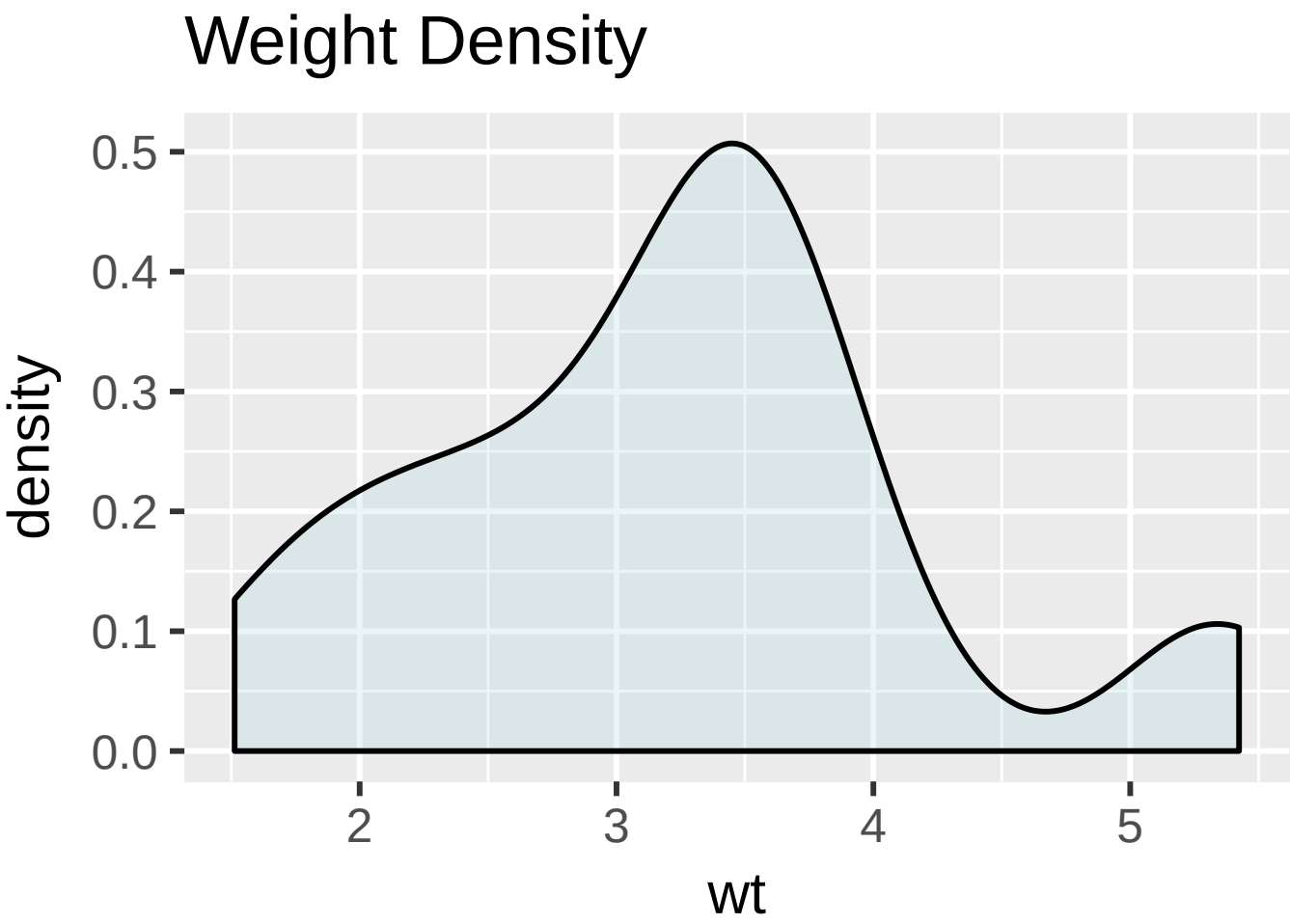
- Individually or in groups, try to recreate the following histograms (3 slides):



Recreate these histograms!!!



Recreate these histograms!!!



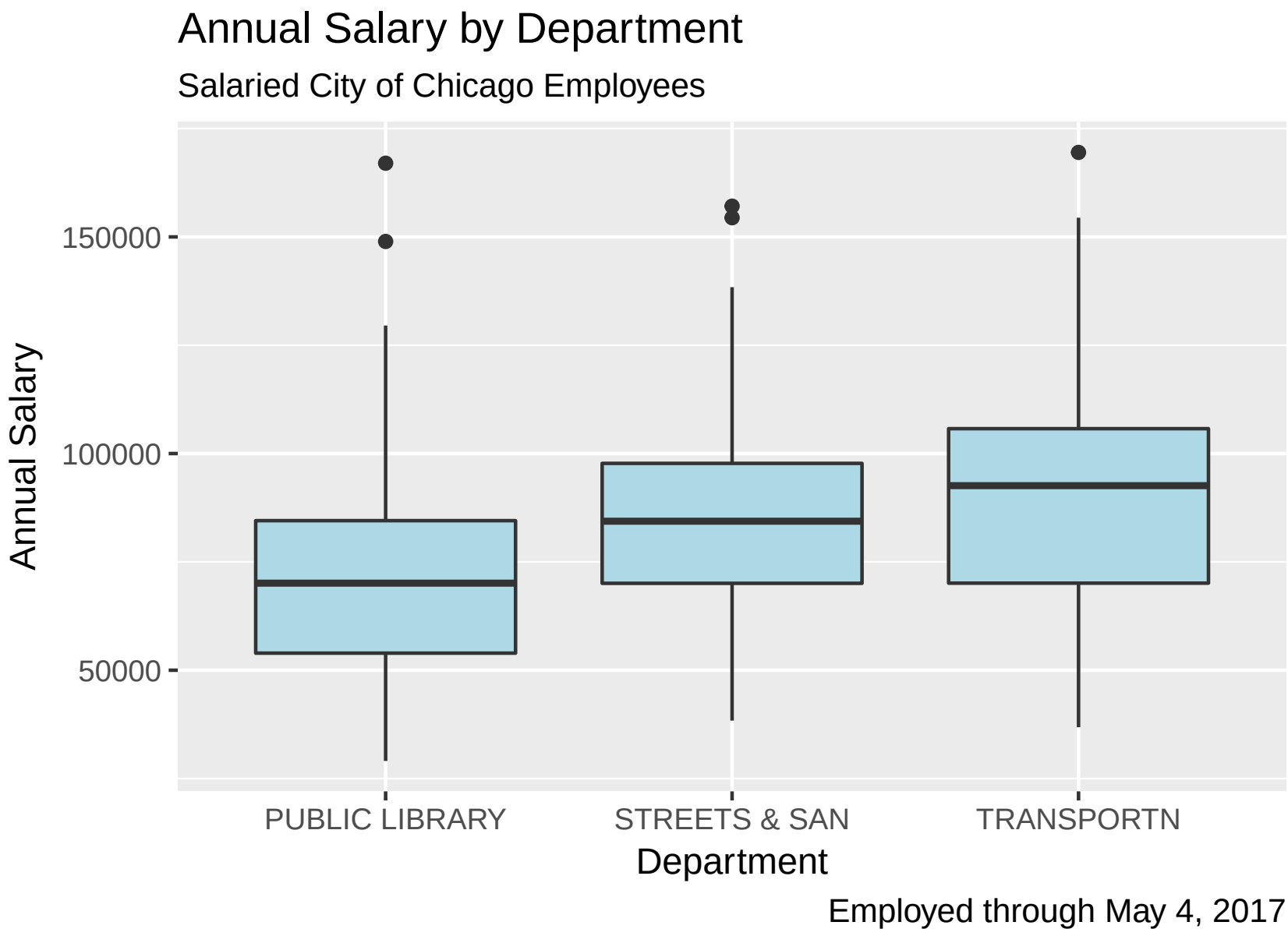
Solutions

```
ggplot(mtcars, aes(x=mpg)) +
  geom_histogram(fill="green", bins=50)

ggplot(mtcars, aes(x=hp)) +
  geom_histogram(fill = "blue", aes(y=..density..), bins=50) +
  ggtitle("Horsepower Density")

ggplot(mtcars, aes(x=wt)) +
  geom_density(alpha = 0.25, fill="lightblue") +
  ggtitle("Weight Density")
```

Box Plots

- Or "Box and Whisker plots," as I learned in elementary school.
- For these, we'll need two aesthetics, x and y. x should be categorical and y should be continuous.
- The box plot will show the median, 1st and 3rd quartiles, range, and any outliers in the continuous data.

Box Plot - Annual Salary by Department, Salaried



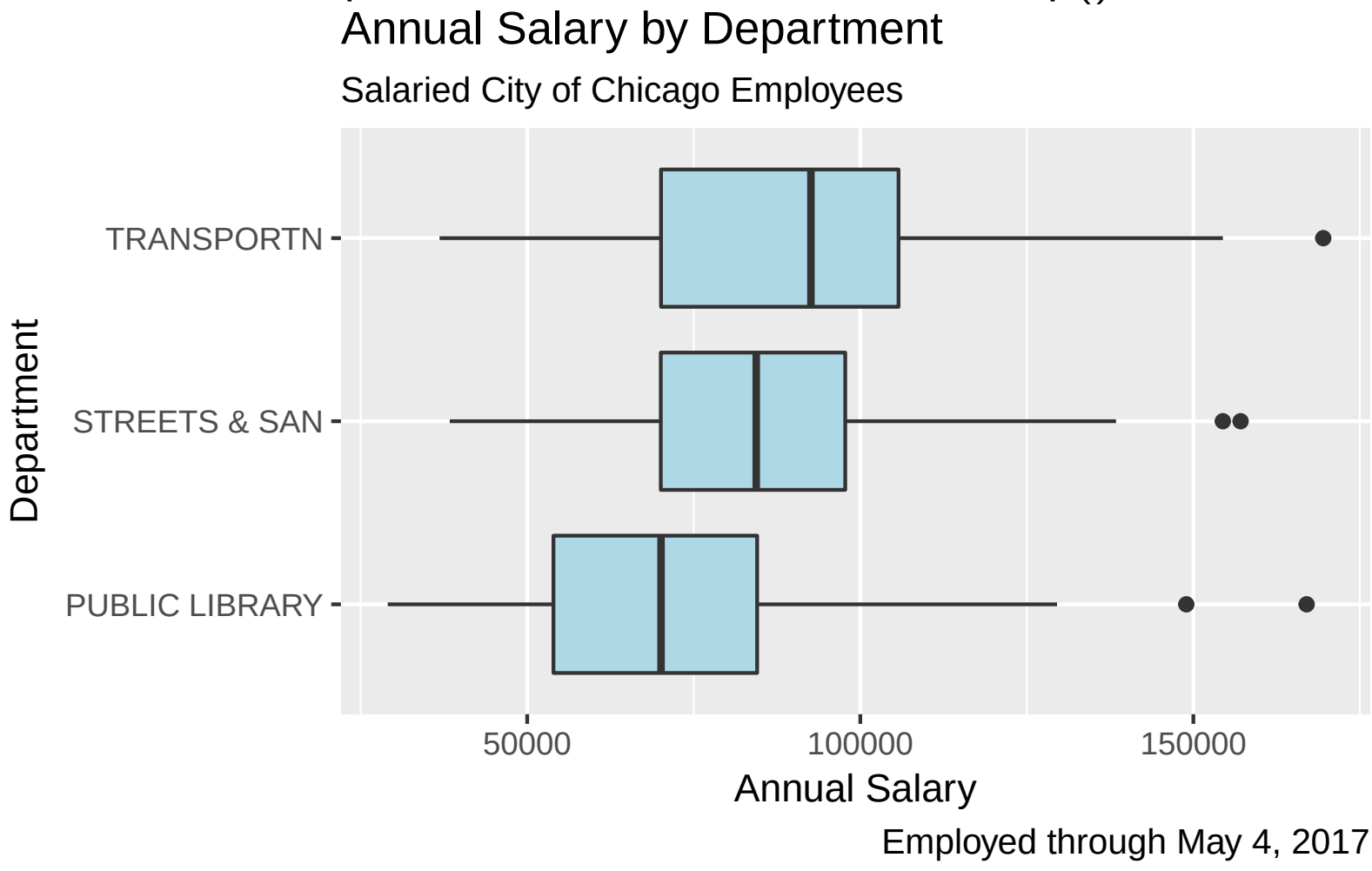
Code for Last Boxplot

- Using only the Salaried employees dataset
- x = categorical, y = continuous
- made it fill lightblue
- added title, subtitle, caption, and axis labels with labs(), which is another way to add these things to any plot!

```
ggplot(chicago_3depts_Salary, aes(x=Department, y=Annual_Salary)) +
  geom_boxplot(fill="lightblue") +
  labs(title = "Annual Salary by Department",
       subtitle="Salaried City of Chicago Employees",
       x="Department", y="Annual Salary",
       caption="Employed through May 4, 2017")
```

Coord_flip

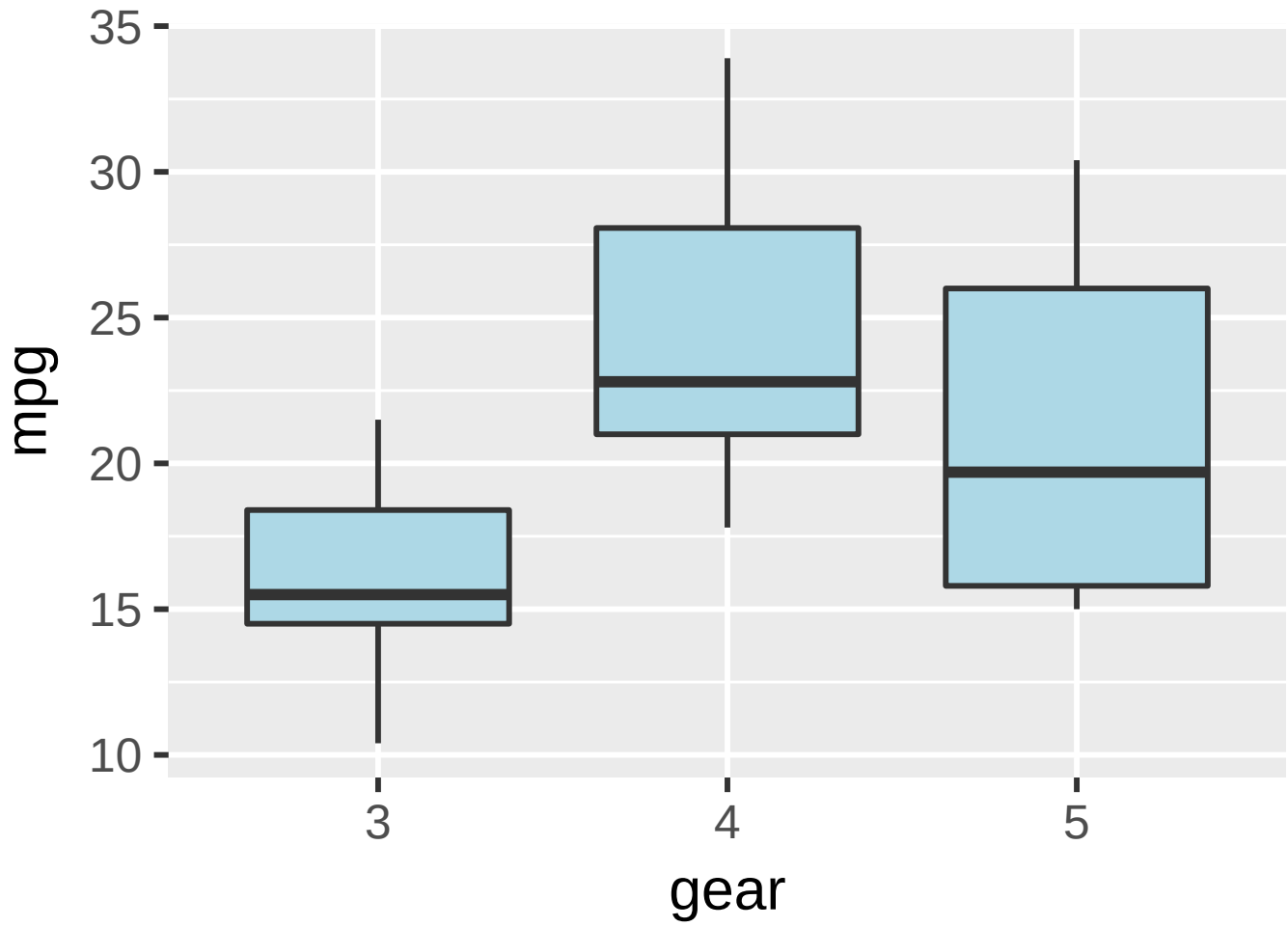
- You can also flip the coordinates with coord_flip()



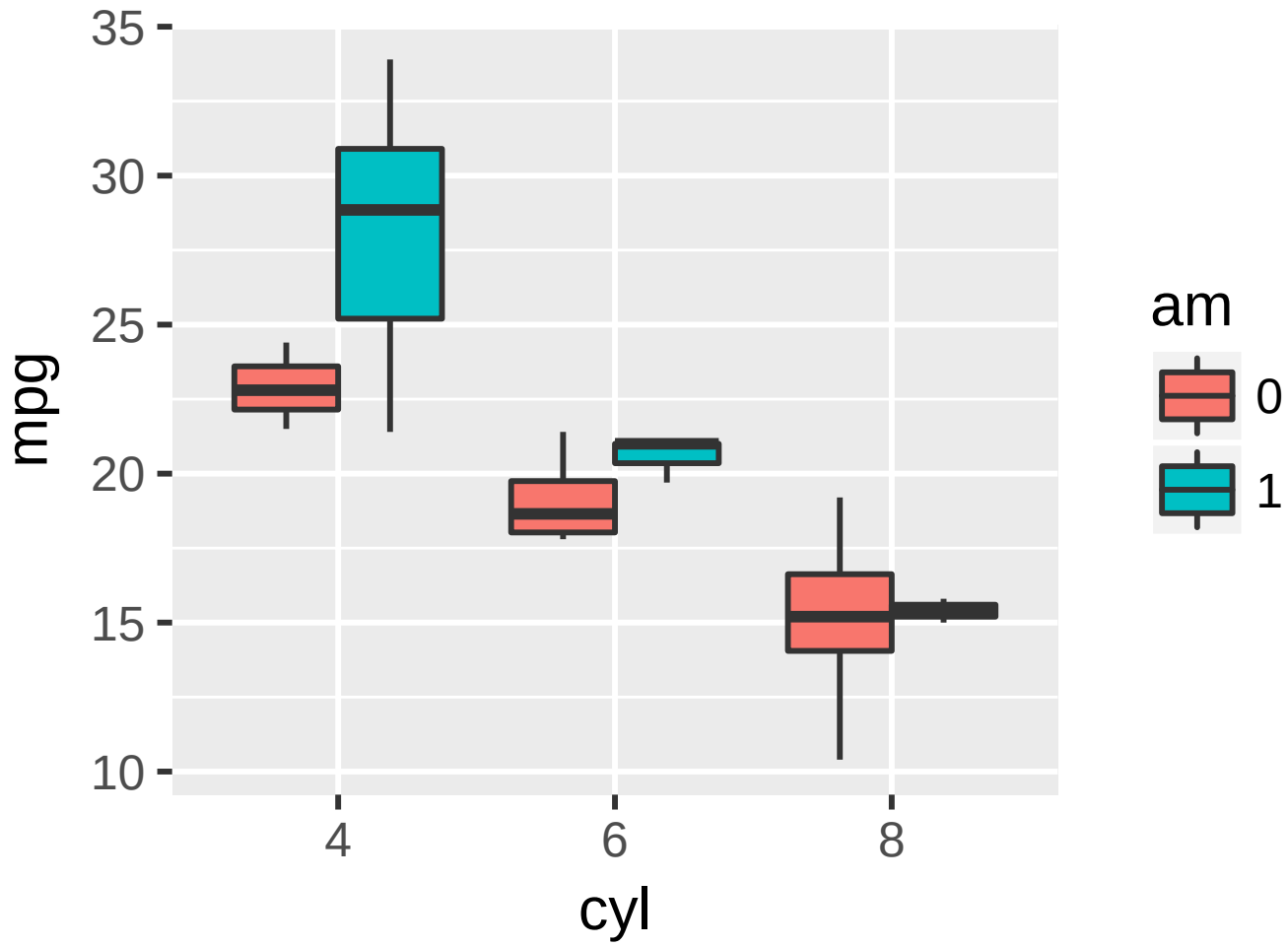
YOUR TURN!

- Like past plots, these boxplots (3 slides) also use the mtcars dataset we've been working with.
- The titles are gone from these plots, so you should look to the axis labels to recreate them successfully.

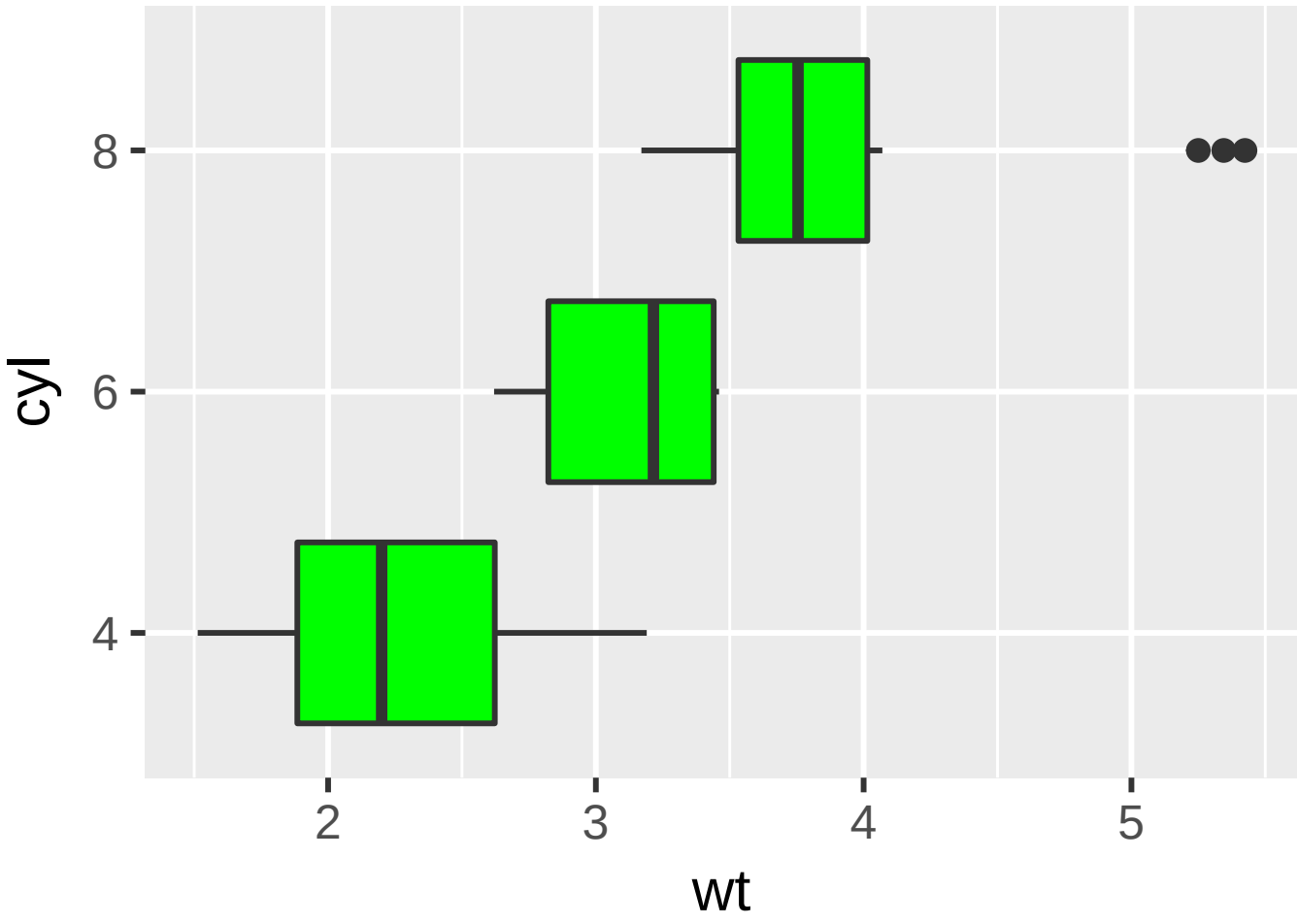
Recreate these boxplots!!!



Recreate these boxplots!!!



Recreate these boxplots!!!



Solutions

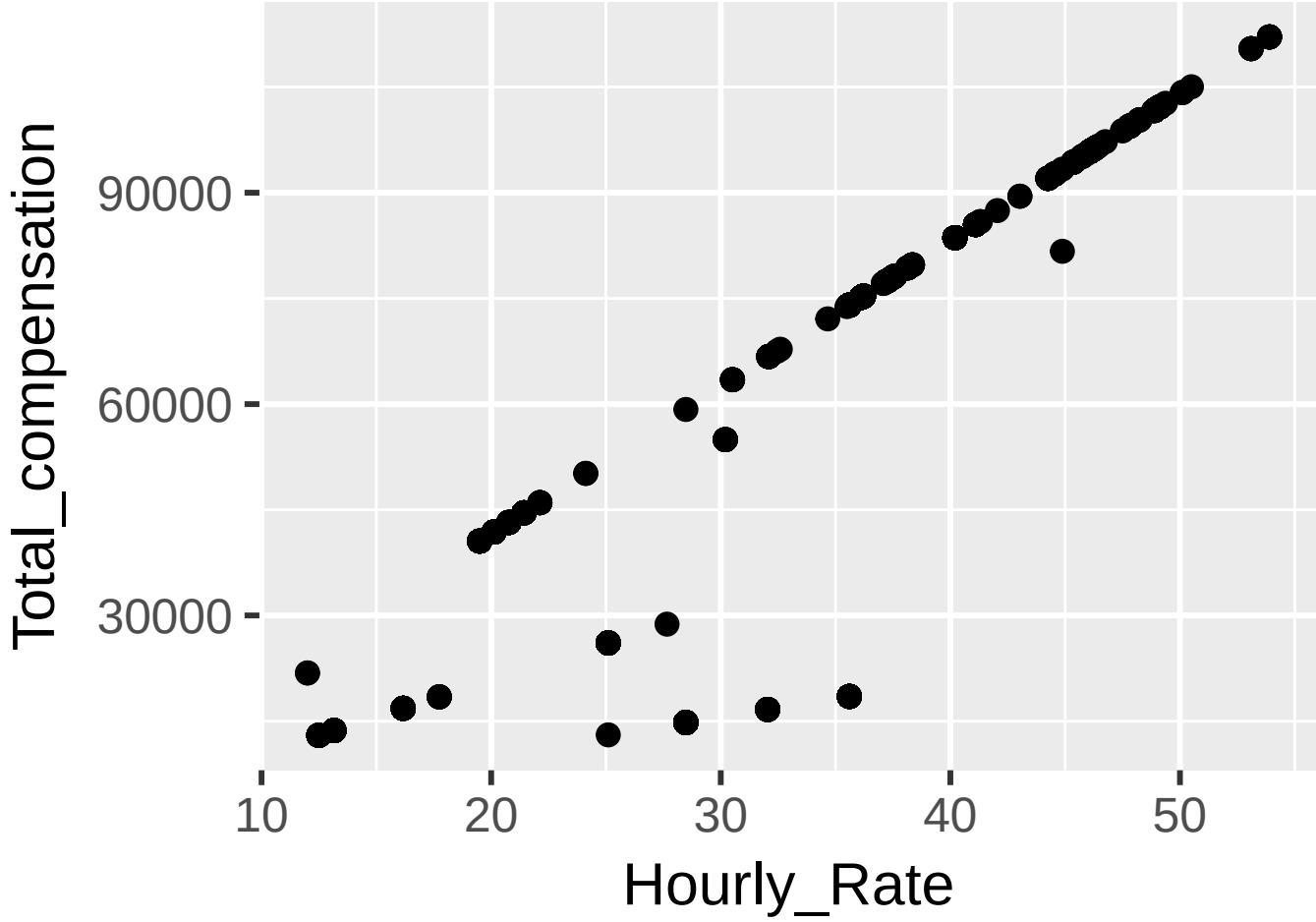
```
ggplot(mtcars, aes(x=gear, y=mpg)) +
  geom_boxplot(fill="lightblue")

ggplot(mtcars, aes(x=cyl, y=mpg, fill=am)) +
  geom_boxplot()

ggplot(mtcars, aes(x=cyl, y=wt)) +
  geom_boxplot(fill="green") +
  coord_flip()
```

Scatterplots

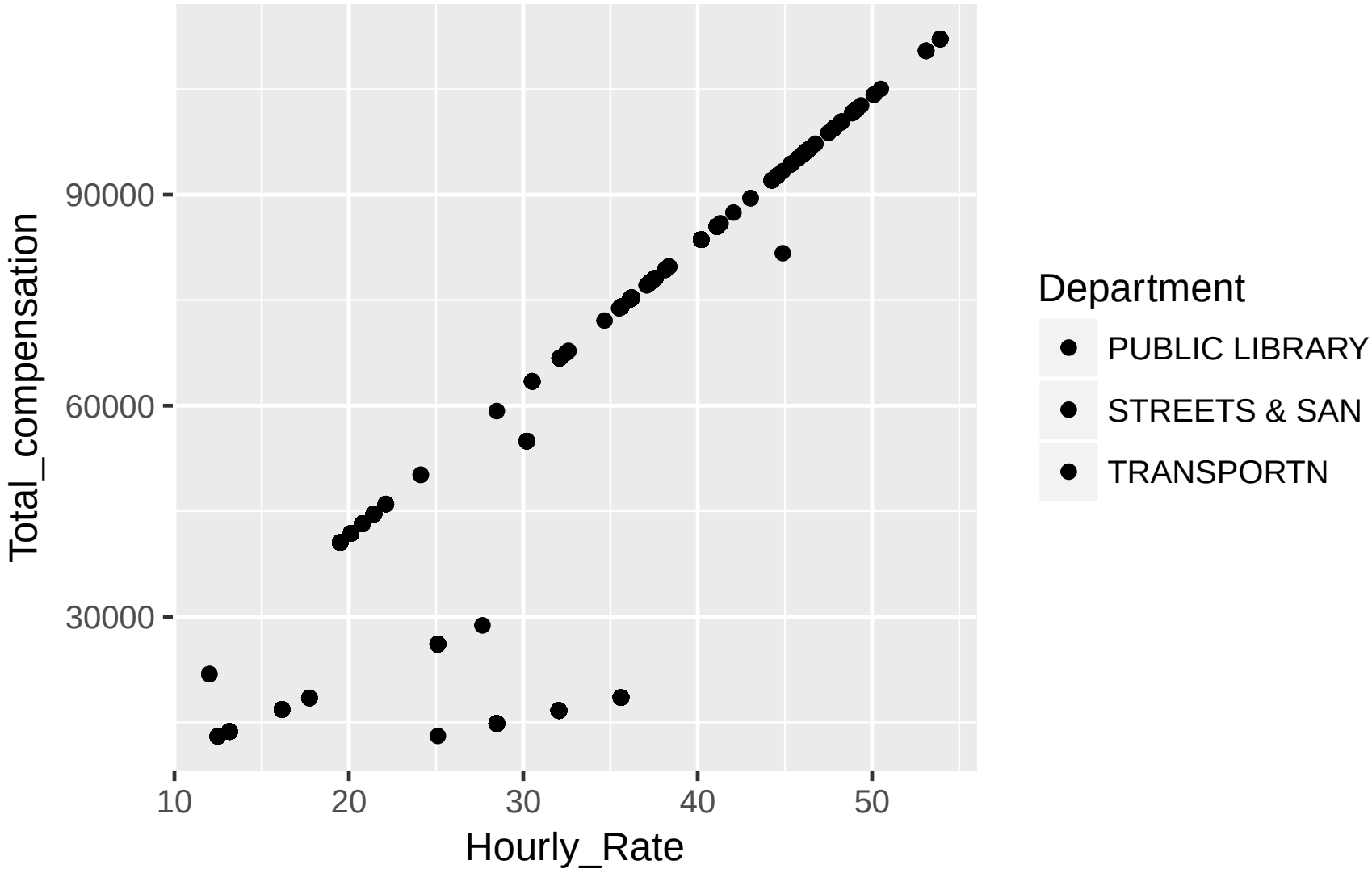
- Acheived with the geom_point()
- Requires x and y, where usually they're both continuous, but x can be categorical. (Unnecessary Editorial commentary: I hate these, even when they're jittered.)
- To be meaninful, scatterplots require a little more TLC than anything I've shown thus far.

Scatterplot e.g.

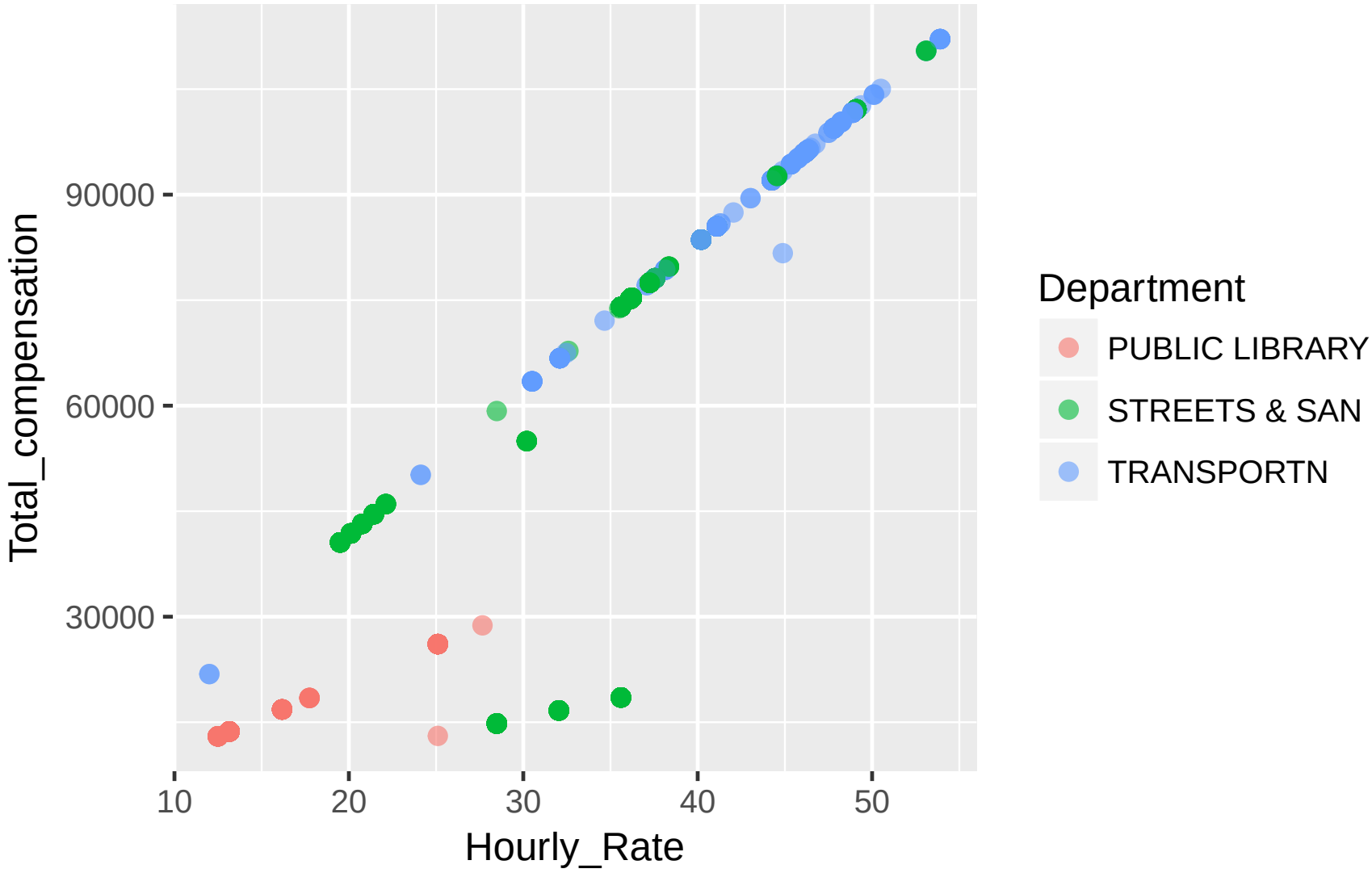
- Let's build an example with the Chicago Hourly employees dataset.
- First, I'll display the plots on this + the next two slides.



Scatterplot e.g.



Scatterplot e.g.



Step-by-Step scatterplots - 1

- First plot is just total compensation by hourly rate.

```
ggplot(chicago_3depts_Hourly,
       aes(x=Hourly_Rate, y=Total_compensation)) +
  geom_point()
```

- However, this isn't that interesting or meaningful. I can't tell anything about those employees who had a total compensation that doesn't fall on the linear relationship we clearly see, and expected

Step-by-Step scatterplots - 2

- So, let's add the fill = call we're used to.

```
ggplot(chicago_3depts_Hourly,
       aes(x=Hourly_Rate, y=Total_compensation, fill=Department)) +
  geom_point()
```

- The legend appears, but notice the color hasn't changed by Department.
- For scatterplots, to get a fill with the color change (helpful!), you have to map the variable to col = (short for color) instead of fill = ...

Step-by-Step scatterplots - 3

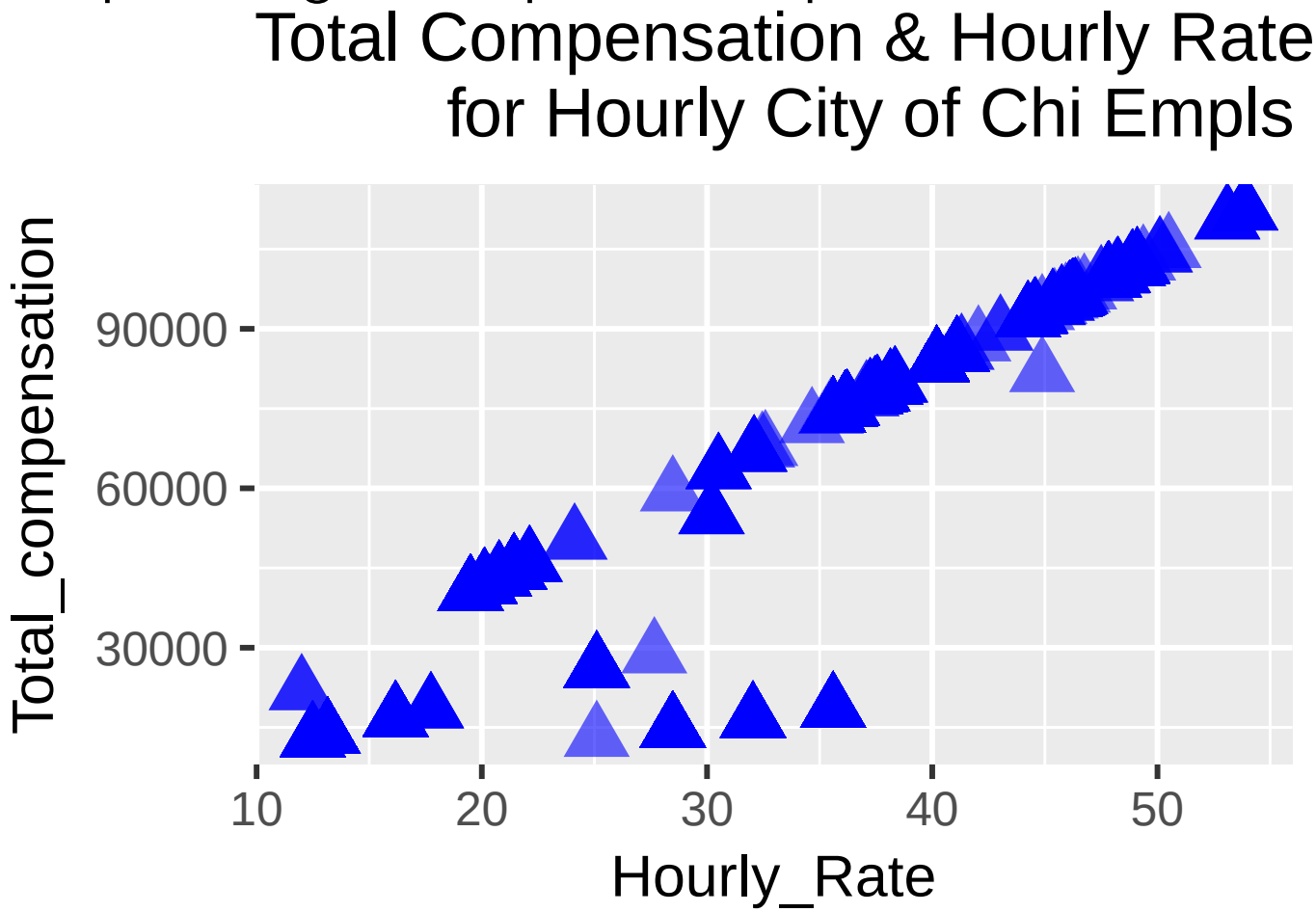
- It will also help to adjust the size (these dots are tiny to my eyes) and the transparency of the points, through alpha = ..

```
ggplot(chicago_3depts_Hourly,
       aes(x=Hourly_Rate, y=Total_compensation, col=Department)) +
  geom_point(alpha=0.6, size=2)
```

- We still don't have a good intuition visually as to why those people's Total compensation is lower, but the plot is more meaningful. (Spoiler alert: all but one are part-time employees - see if you can create a graph showing this!)

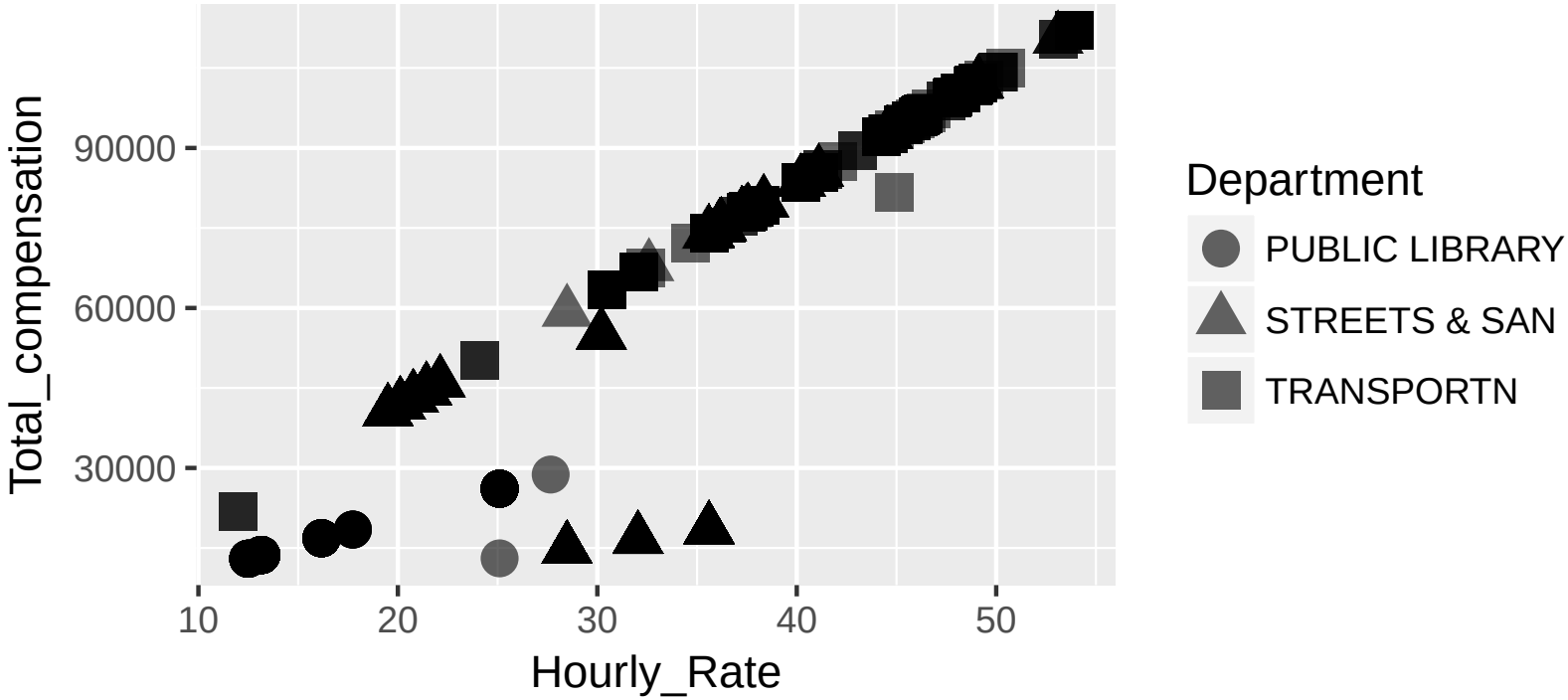
Change the Shapes

- Simple change of shape with shape = 17



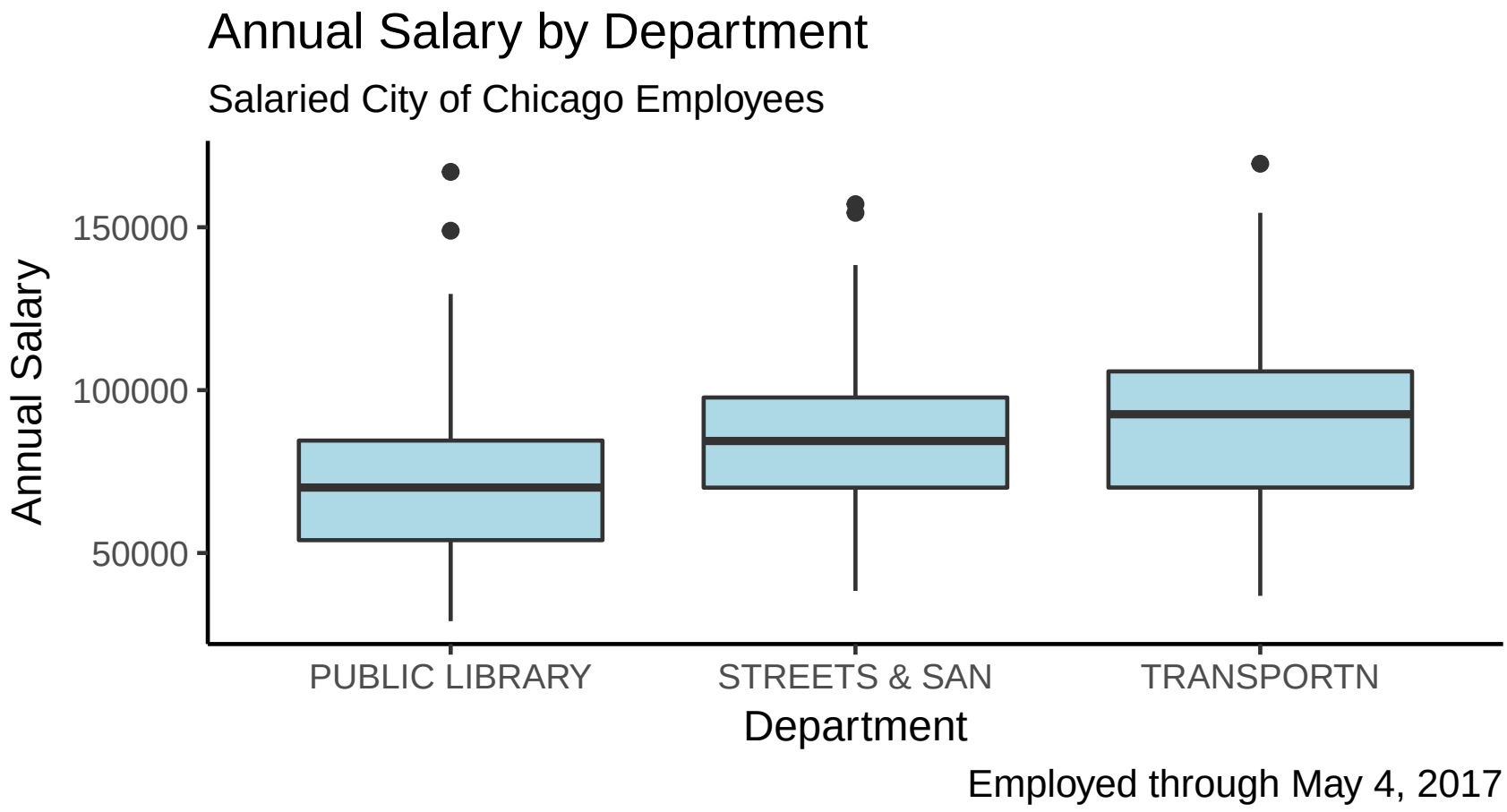
Change the Shapes

- Mapping the shape change to the Department variable, though usually in scatterplots this isn't recommended (you can see it's difficult to see the differences between the plotted shapes, even if I changed the colors/transparency.)



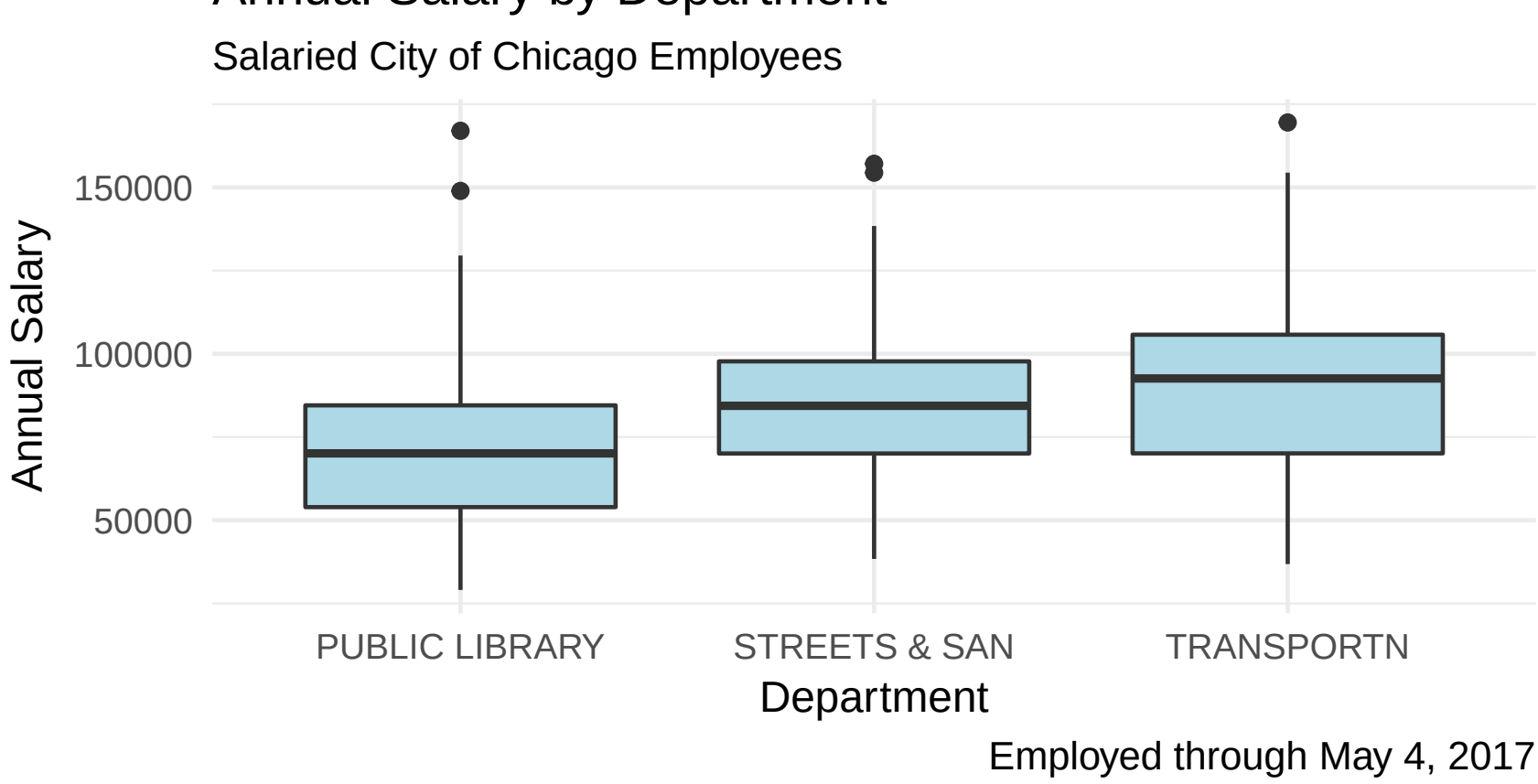
Change the Theme

- Basic theme calls, like theme_classic() as shown, change the background of the plot.



Change the Theme

-Another option that changes the background is theme_minimal().



Save your plots

- You can save plots a few different ways.
- Export directly from the viewer
- or use ggsave(), a built-in from ggplot. The code below will save a 4 by 4.5 inch plot to your working directory
- You can see which folder is your wd with getwd(), and set a new one with setwd("place").

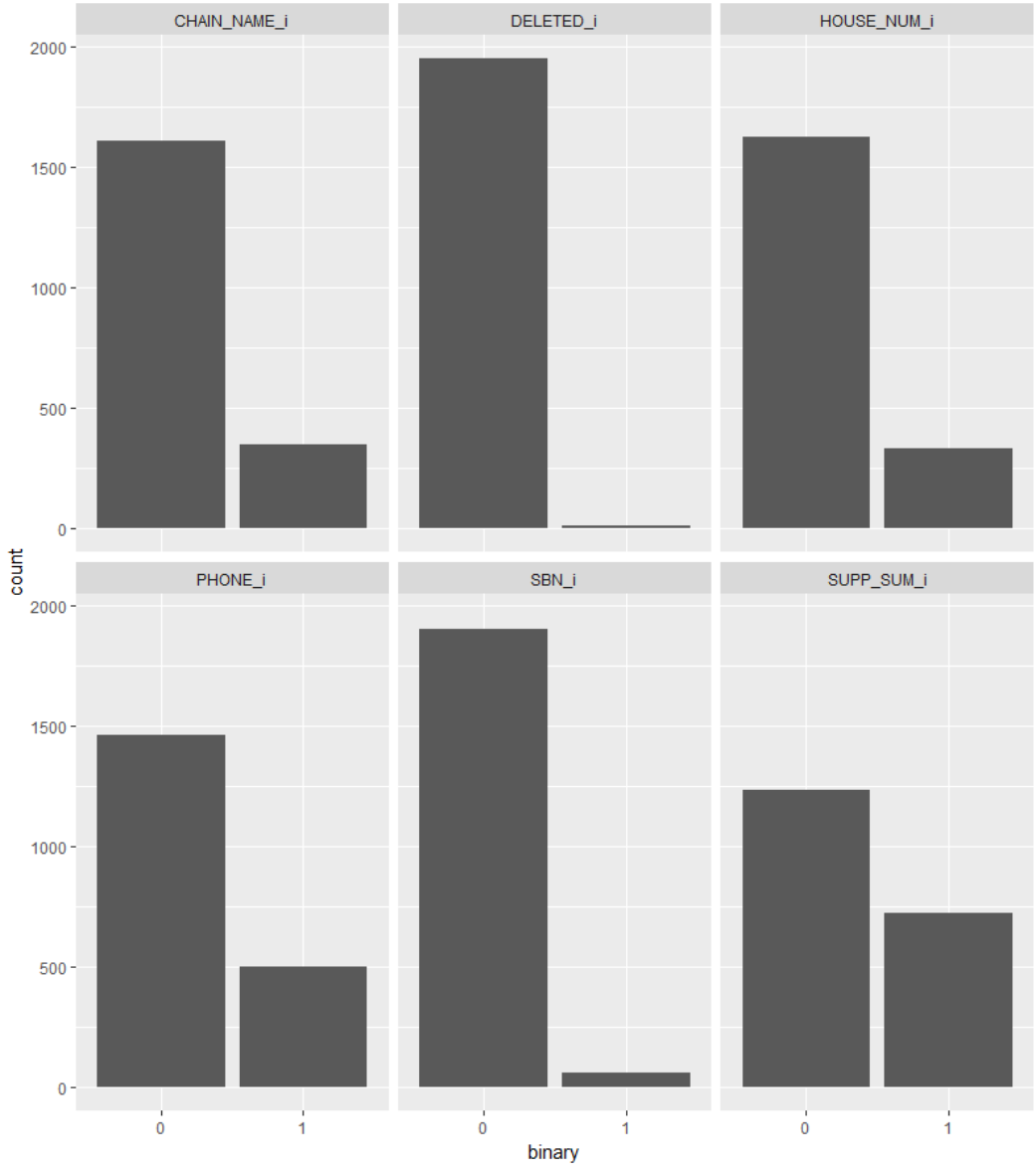
```
ggsave("plot.png", width=4, height=4.5)
```

Miscellania

- Use the 'Data Viz with ggplot2' cheat sheet, available here: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
- I barely touched Stats or Scales, and only flirted with Coordinate Systems and Faceting.
- I don't use jittering very often so it's not covered here.
- Give your plots more meaningful titles that my example titles used throughout.
- I learned ggplot by doing. I had graphs I needed for work or a project, I googled "how do you graph this thing," and followed tutorials.

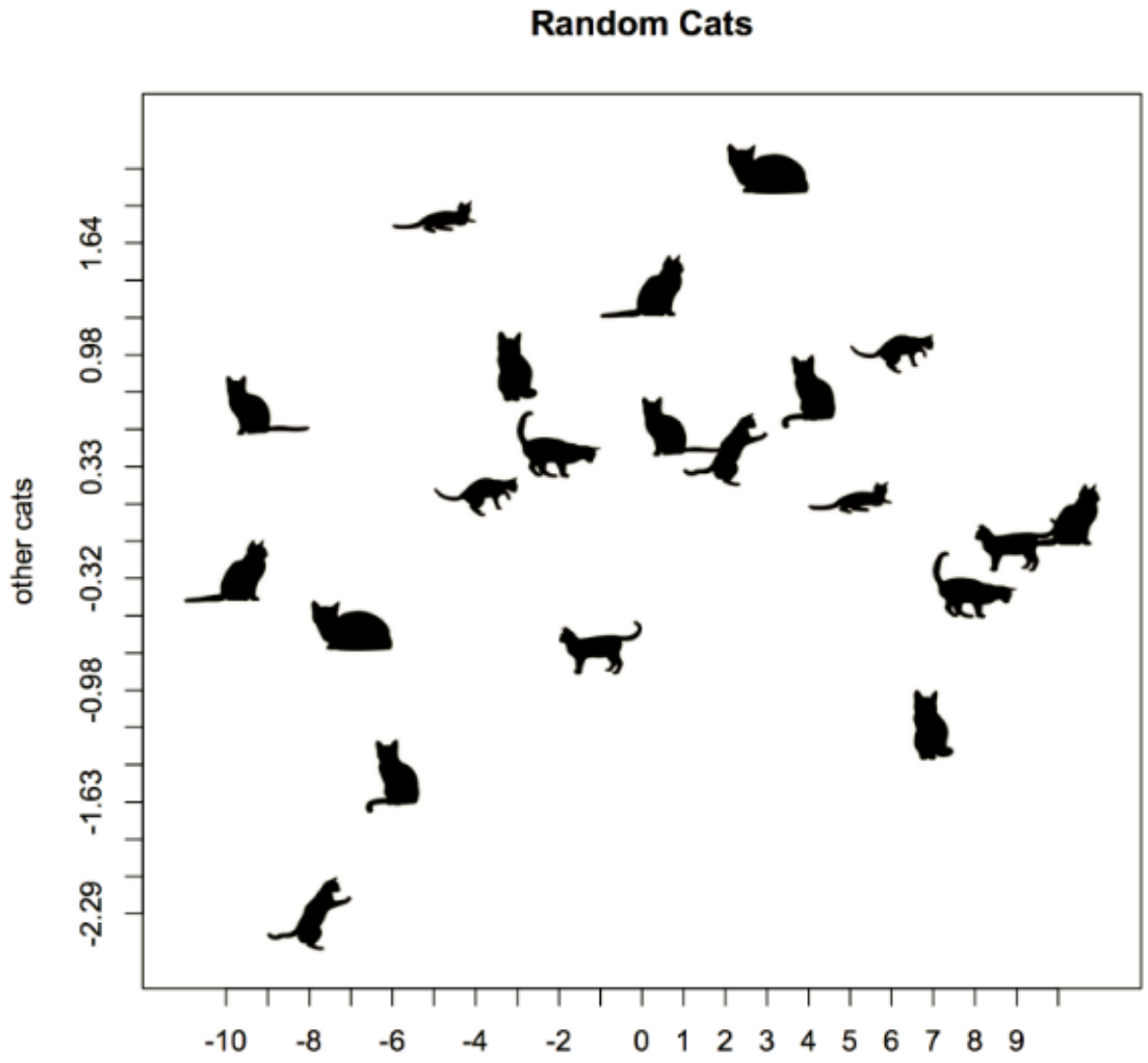
The sky is the limit!

- An example from my work:



The sky is the limit!

- catterplots aren't actually ggplots, but are fun. Install the CatterPlot package and have fun.
- Found here: https://github.com/GibbsdavidI/CatterPlots



Contact me:

- on Twitter @kaelen_medeiros
- on slack under this name, Kaelen Medeiros
- kaelenmedeiros at gmail dot com
- after this talk I can answer more questions too!