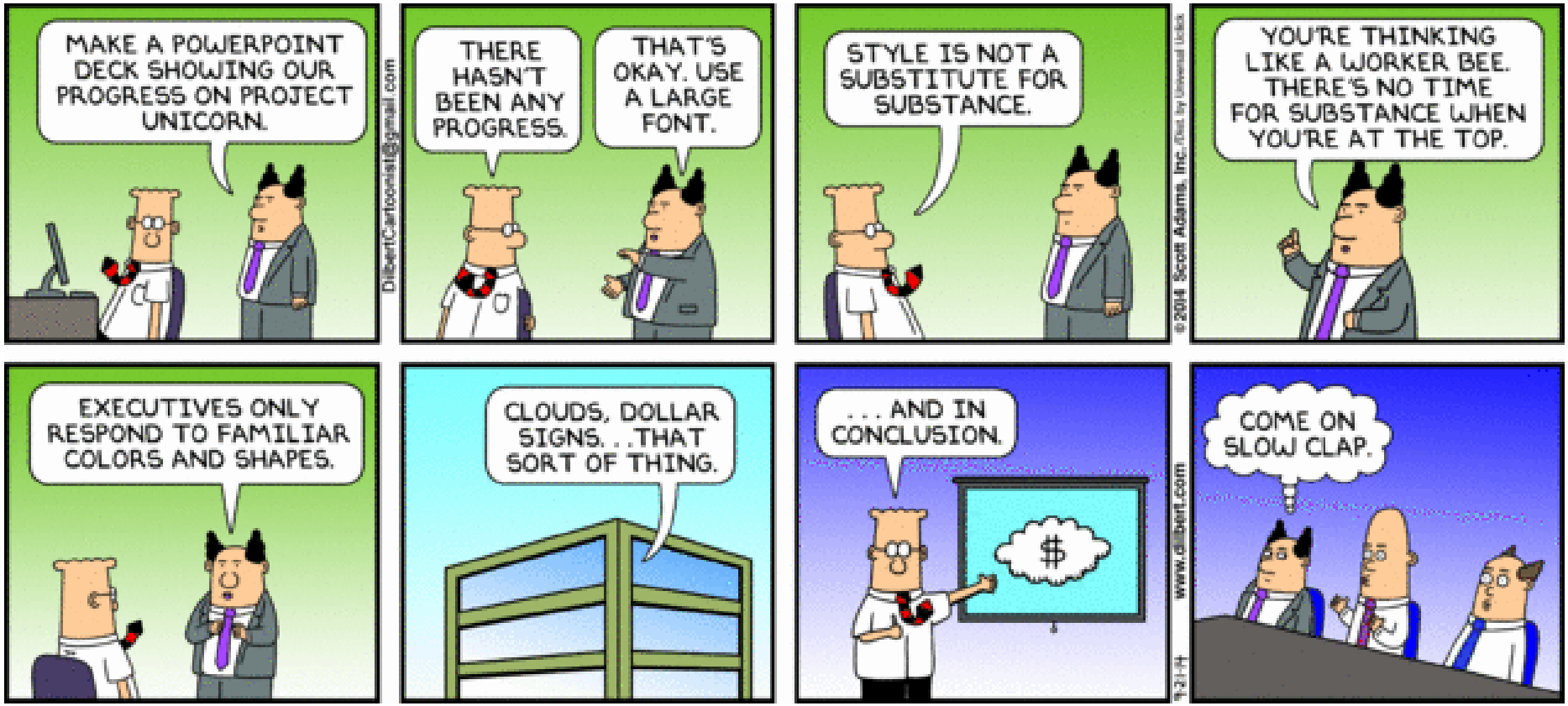

MAKE A POWERPOINT DECK SHOWING OUR PROGRESS ON PROJECT UNICORN.
DilbertCartoonist@gmail.com
THERE HASN'T BEEN ANY PROGRESS.
THAT'S OKAY. USE A LARGE FONT.
STYLE IS NOT A SUBSTITUTE FOR SUBSTANCE.
© 2014 Scott Adams, Inc. /Dist. by Universal Uclick
YOU'RE THINKING LIKE A WORKER BEE. THERE'S NO TIME FOR SUBSTANCE WHEN YOU'RE AT THE TOP.
EXECUTIVES ONLY RESPOND TO FAMILIAR COLORS AND SHAPES.
CLOUDS, DOLLAR SIGNS. . .THAT SORT OF THING.
. . . AND IN CONCLUSION.
$
www.dilbert.com
1-31-14
COME ON SLOW CLAP.

DCMB BioComputing BootCamp
Day 3, Lecture 3:
Data Exploration and Visualization in R

Armand Bankhead

bankhead@umich.edu

8/21/2019

Now What?

- Scenario:
  - You understand the fundamentals of R
  - You've read your data into an R data frame
- During this session we will talk about

1. Basic data summarization
2. Visualize data with plots

Overview

1. Summarizing Data in R
2. Creating Plots in R Using ggplot2
    a. Histograms
    b. Boxplots
    c. Scatter plots

Example Data Set:

Cancer Research

Home About Articles For Authors Alerts News

Molecular and Cellular Pathobiology

Activation of Wnt/β-Catenin in Ewing Sarcoma Cells Antagonizes EWS/ETS Function and Promotes Phenotypic Transition to More Metastatic Cell States

Elisabeth A. Pedersen, Rajasree Menon, Kelly M. Bailey, Dafydd G. Thomas, Raelene A. Van Noord, Jenny Tran, Hongwei Wang, Ping Ping Qu, Antje Hoering, Eric R. Fearon, Rashmi Chugh, and Elizabeth R. Lawlor

DOI: 10.1158/0008-5472.CAN-15-3422 Published September 2016

- Ewing's sarcoma: rare bone and soft tissue cancer occurring in children and teenagers
  - 70-80% survival
- *In vitro* CHLA25-7TGP ES cells stimulated to over-express WNT3A
- RNA-Seq profiling used to quantify gene expression

Download pedersenLog2RPKM_v1.txt and pedersenLog2_matrix_v1.txt from the Day3 course website.

Exercise: Write a Script to Read Pedersen Gene Expression Data into a Data Frame

In-Class Exercise: Try It Out!

1. Download both Pedersen data files
2. Use setwd() to move to the data file folder
3. options(stringsAsFactors=F)
4. Use the read.delim() function to read in "pedersenLog2RPKM_v1.txt" file into a data frame called "data1"
5. Use the head() and dim() function to find out about the structure of this data file

How many rows? What are the columns?

In-Class Exercise: Try It Out!

Quickly Calculate Simple Statistics

- R has many built in statistical functions that use fast vector and matrix operations
- No need to write a for loop, sum, and then divide by n
- Just provide a vector of data to the mean() function:

```
> mean(data1$log2RPKM)
```

- With one line of code you have take the mean of 97,000 values!

Exercise: Use mean(), median(), max(), min(), summary() functions on the Pedersen data

Using the table() Function

- When exploring new datasets it is often useful to count the number of values
- table() can be used to build a contingency table of the counts of each value
  - For one column:

```
> table(data1$tx)

control   WNT3A
  48585   48585
```

  - For multiple columns:

```
> table(data1$tx,data1$rep)

             1     2     3
control 16195 16195 16195
WNT3A   16195 16195 16195
```

Using the aggregate() Function

In-Class Exercise: Try It Out!

- Often times we want to perform a function on subsets of our data
  - example question: What is the mean expression for each sample?
- aggregate() splits data into subsets, computes summary statistics for each and returns the result
- aggregate() takes several arguments:

formula input data

```
> aggregate(log2RPKM ~ sample, data1, FUN='mean')
        sample log2RPKM
1 control_rep1 2.598991
2 control_rep2 2.583161
3 control_rep3 2.579987
4   WNT3A_rep1 2.593578
5   WNT3A_rep2 2.583571
6   WNT3A_rep3 2.598349
```

summary statistic

Exercise: Use aggregate() to calculate the maximum log2RPKM value per sample.

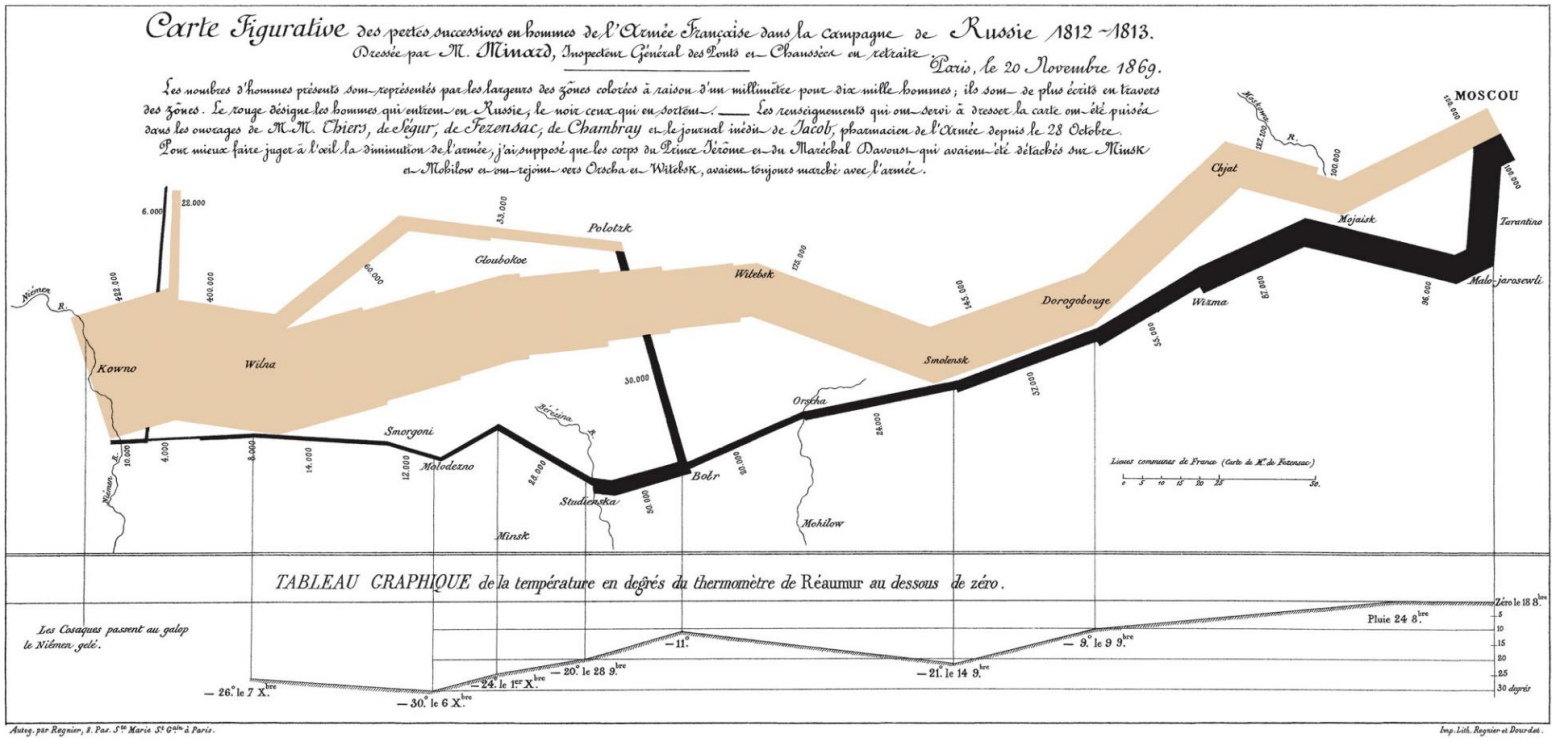
formula format: y ~ x1

- **y is a numeric value**
- **x1 is a grouping variable**
- **possible to specify multiple groups as x1 + x2 + …**

Overview

1. Summarizing Data in R
2. **Creating Plots in R Using ggplot2**
    a. Histograms
    b. Boxplots
    c. Scatter plots

Data Visualization Allows Researchers to Visually Present Data



- Minard's 1869 diagram of Napoleonic France's invasion of Russia
  - Line width indicates size of army
  - Color indicates army's course to and from Russia

Data Visualization Allows Researchers to Visually Present Data

- Data visualizations should:
  - Show the data
  - Avoid distorting the data
  - Present many numbers in a small space
  - Make large data sets coherent
  - Serve a reasonably clear purpose
  - Be closely integrated with the statistical and verbal descriptions of a data set

https://en.wikipedia.org/wiki/Data_visualization

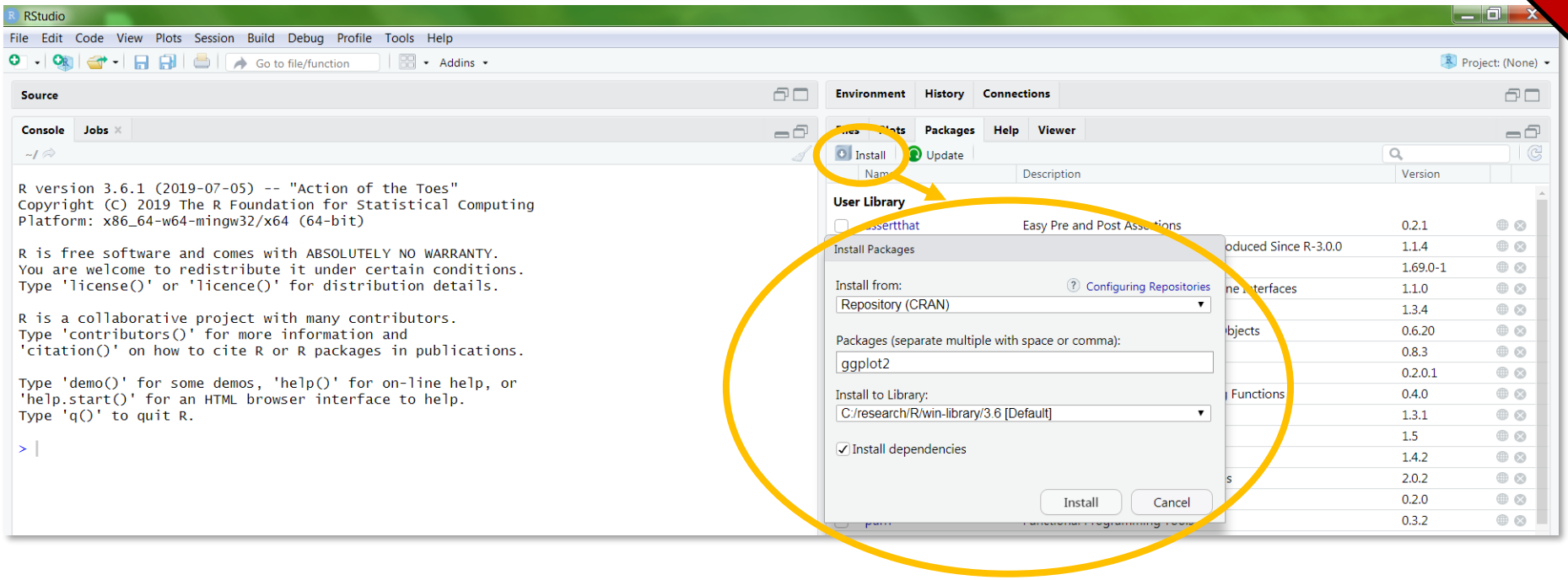
R Base Graphics Versus ggplot2

- R comes with “base graphics” built in to support commonly used data visualizations
- Today we will focus on using an alternative data visualization framework called ggplot2
- ggplot2 is an external package that must be downloaded, installed, and loaded with the library command
- A common practice is to use ggplot2 to construct publication quality graphs but still use base graphics to quickly visualize data

https://flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics/

In-Class Exercise:
Try It Out!

Install ggplot2 Package

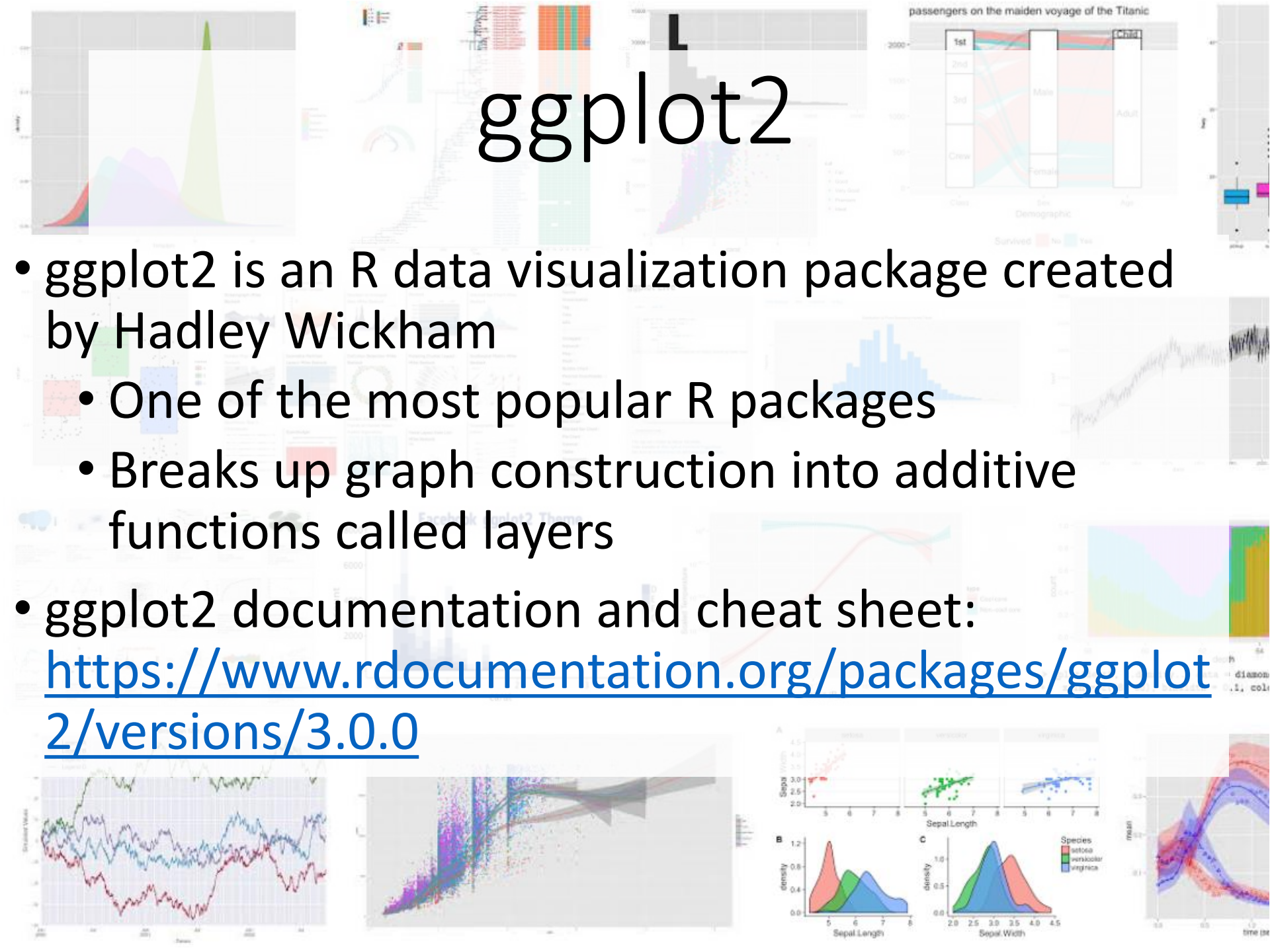


To install ggplot2:

1. select “packages” tab
2. select “Install”
3. Type in ggplot2
4. Select “Install”

ggplot2

- ggplot2 is an R data visualization package created by Hadley Wickham
  - One of the most popular R packages
  - Breaks up graph construction into additive functions called layers
- ggplot2 documentation and cheat sheet: https://www.rdocumentation.org/packages/ggplot2/versions/3.0.0



Creating a Visualization with ggplot2

- ggplot2 visualization function calls consist of several basic components

1. ggplot()
2. geom_XXX()
3. optional layers
4. ggsave()

- Multiple function calls are combined together using "layers"
- aes(thetic) functions are used to map input data to plot features (e.g. x axis, y axis, colors)

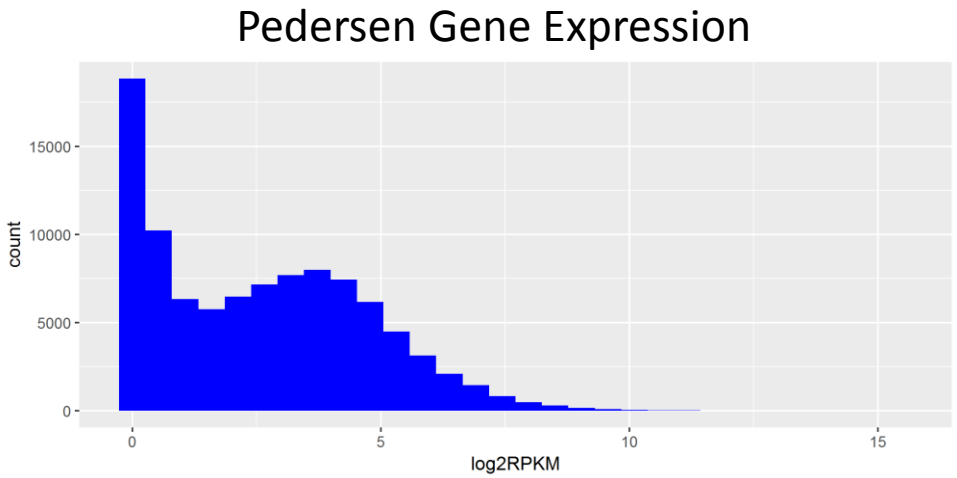
```
options(stringsAsFactors=F)

library(ggplot2)

inFile = 'data.txt'
data1 = read.delim(inFile)

ggplot(data1,aes(x = log2RPKM)) +
 geom_histogram()
ggsave('histogram.png')
```

input data frame

aesthetic

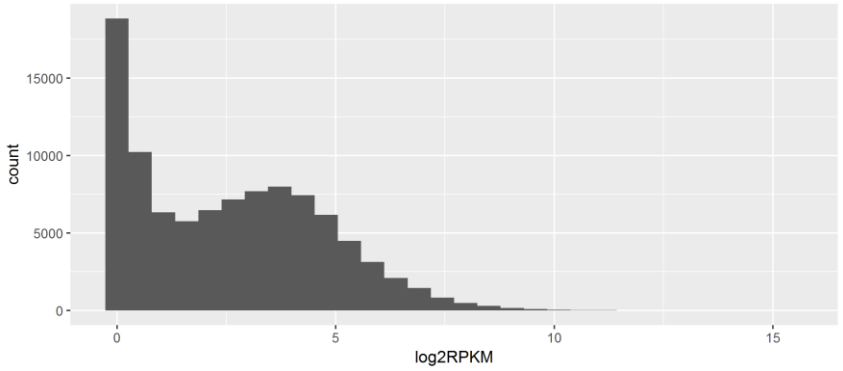
Visualizing Data Using Histograms

- histogram: a type of bar graph visualization in which data measurements are counted based on value
  - For **discrete** measures it shows the frequency of values in each category
  - For **continuous** measure it shows the frequency of values occurring in small intervals covering the whole range



How to Create a Basic Histogram Using ggplot2

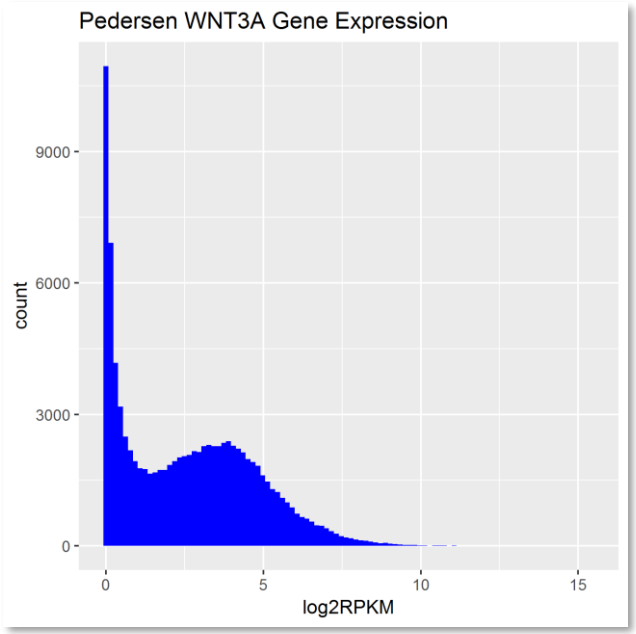
```
options(stringsAsFactors=F)

library(ggplot2)

inFile1 =
'pedersenLog2RPKM_v1.txt'
data1 = read.delim(inFile1)

ggplot(data1,aes(x = log2RPKM)) +
  geom_histogram()
ggsave('histogram1.png')
```



In-Class Exercise: Try It Out!

Exercise: Create a histogram using the code from the previous page and update your visualization to:

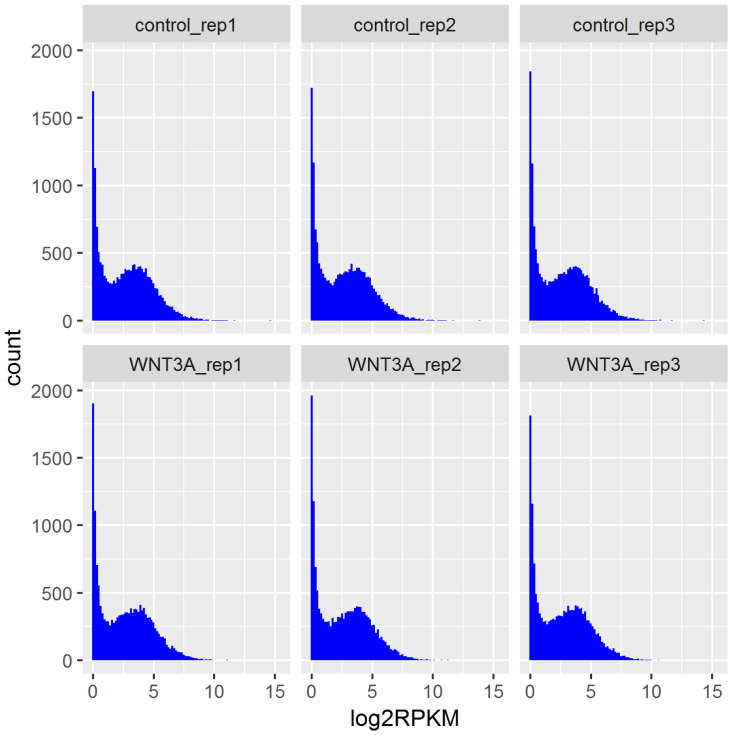
1. Change the color
   - HINT: ?geom_histogram
   - HINT: fill = "blue"
2. Set the image width and height to be 5 inches
   - HINT: ?ggsave
   - HINT: height = 5
3. Adjust the number of bins to 100
4. Add a title
   - HINT: ?labs



In-Class Exercise:
Try It Out!

Create Sub-plots Using Facets

- Sub-plots can easily be created using facet layers:
  - facet_wrap()
  - facet_grid()



Exercise: Update your histogram visualization to facet on sample

1. Add a facet_wrap() layer
   - HINT: + facet_wrap(~sample)

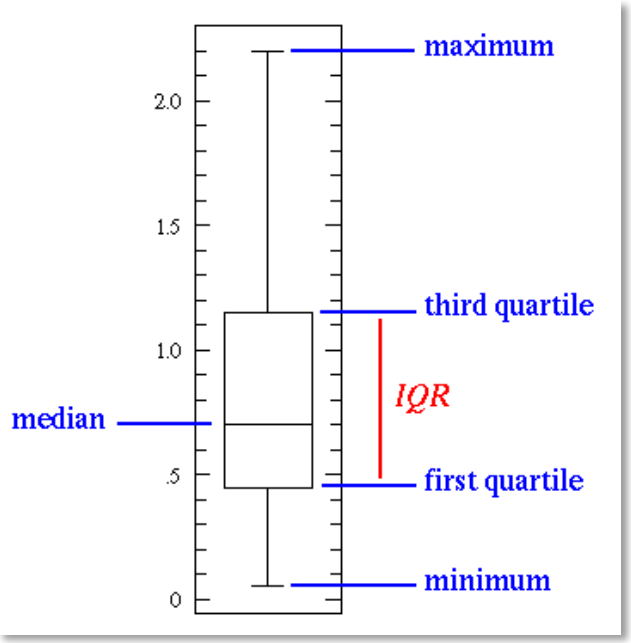
Overview

1. Summarizing Data in R
2. Creating Plots in R Using ggplot2
    a. Histograms
    b. Boxplots
    c. Scatter plots

Visualizing Data Using Boxplots

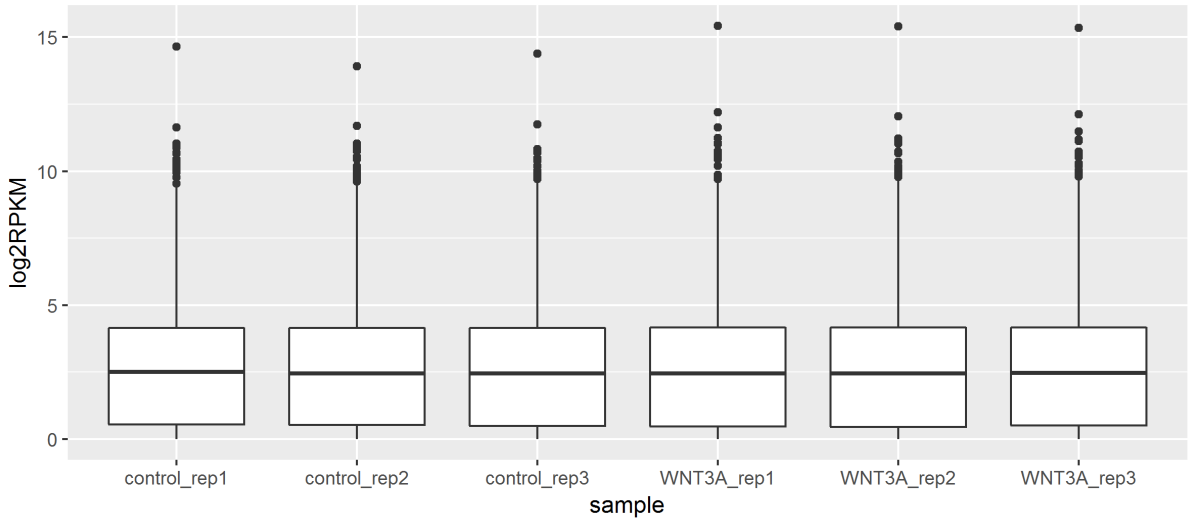
- boxplot: graphically represents data distributions using quartiles
- box-and-whisker plot: includes boxplots with lines extending from boxes to indicate variability outside the upper and lower quartiles

- Why it is useful?
  - Summarize the main characteristics of the data: Mean/median, quartile, spread, symmetry and outliers.
  - Efficient – less complicated than histogram
  - Allows us to represent multiple data distributions in the same graph



How to Create a Boxplot Using ggplot2

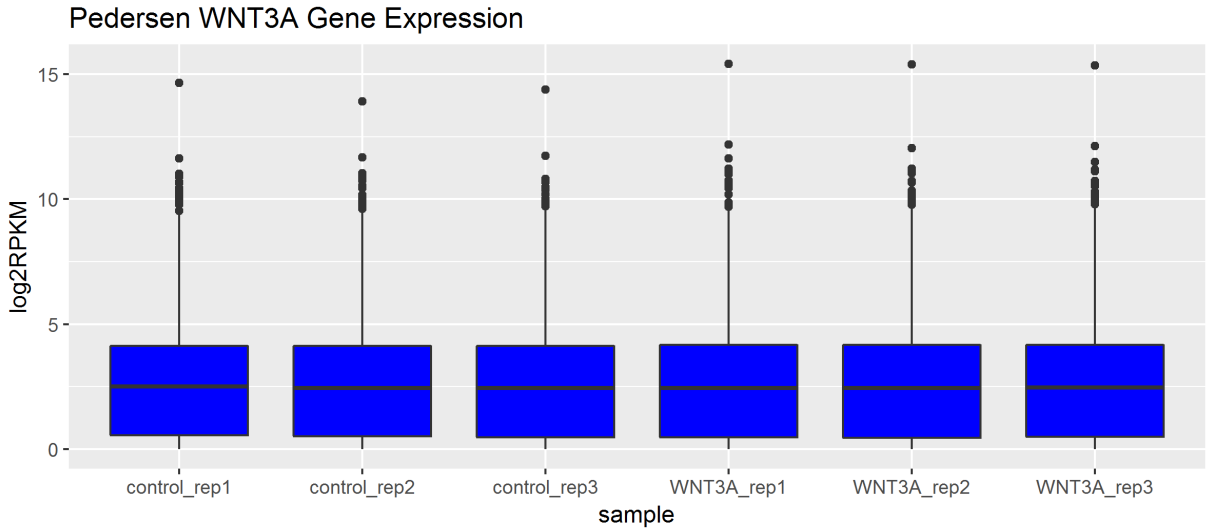
```
options(stringsAsFactors=F)

library(ggplot2)

inFile1 = 'pedersenLog2RPKM_v1.txt'
data1 = read.delim(inFile1)

ggplot(data1,aes(x = sample, y = log2RPKM)) +
  geom_boxplot()
ggsave('boxplot1.png')
```



In-Class Exercise: Try It Out!

Exercise: Create a boxplot using the code from the previous page and update your visualization to:

1. Change the fill color
   - HINT: fill = 'blue'
2. Add a title
   - HINT: ?labs

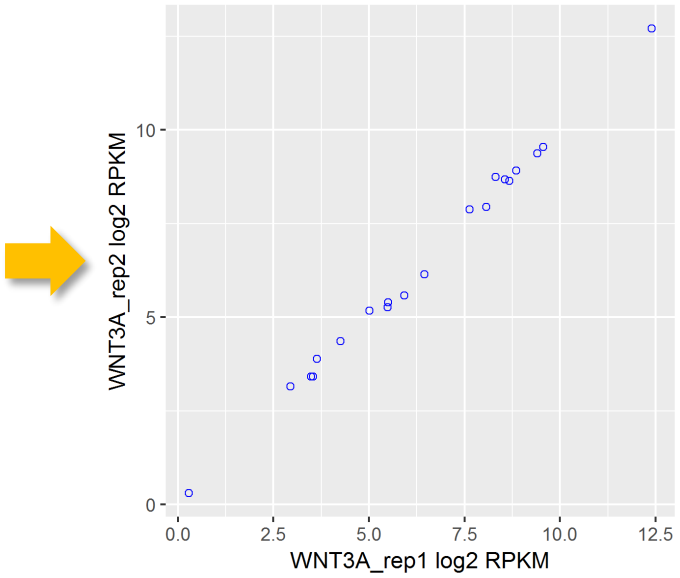


If time: Try creating a violin plot using geom_violin()

Visualizing Data Using Scatter Plots

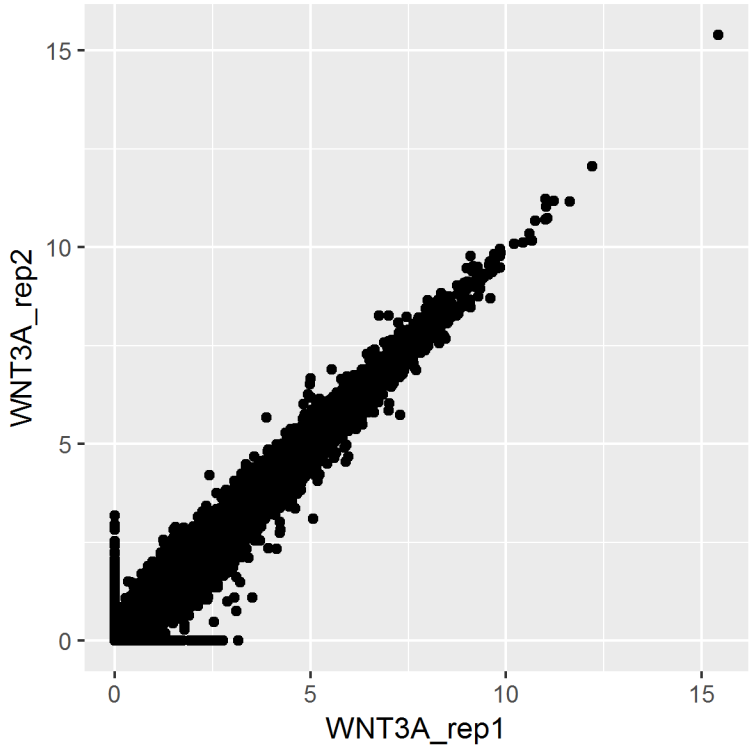
- Scatter Plots are visualizations that display two data values for the same measurement
  - example: two sample replicates expression values for each gene
- Data points that are not on the diagonal indicate disagreement
- We expect strong agreement between sample replicates

gene	WNT3A_rep1	WNT3A_rep2
A1BG	2.38	1.64
A1BG-AS1	0.83	0.58
A1CF	0.02	0.00
A2M	0.30	0.67
A2M-AS1	0.09	0.10
A2ML1	0.33	0.73
A2MP1	0.00	0.00
A4GALT	4.20	4.82
A4GNT	0.00	0.00
AAAS	5.57	5.53
AACS	2.70	2.52
AACSP1	0.13	0.15
AADAC	0.12	0.00
AADACL2	0.00	0.00
AADACL4	2.30	2.03
AADAT	0.94	1.19
AAED1	1.20	1.51
AAGAB	4.36	4.29
AAK1	1.68	1.97
AAMDC	4.41	3.78



How to Create a Scatter Plot Using ggplot2

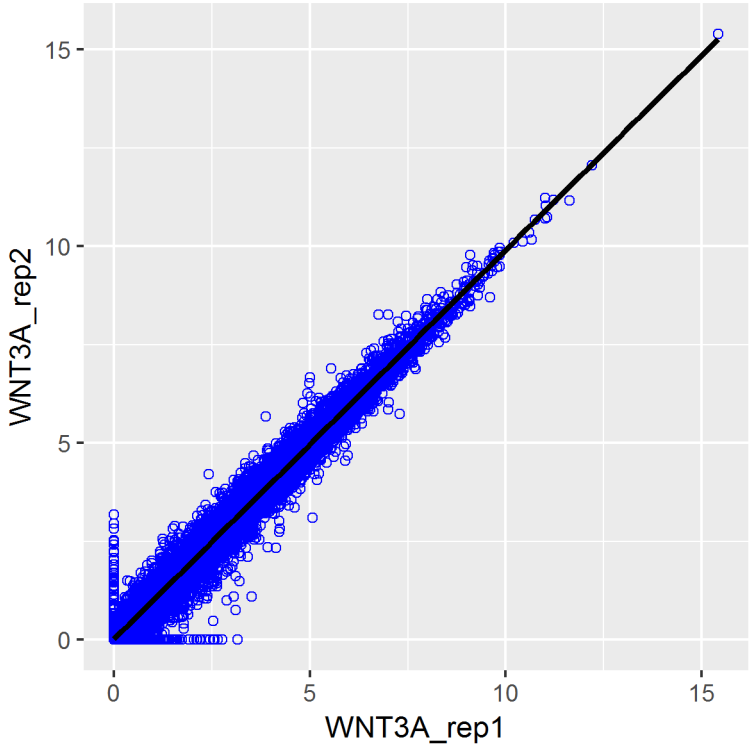
```
options(stringsAsFactors=F)

library(ggplot2)

inFile2 =
'pedersenLog2RPKM_matrix_v1.txt'
data2 = read.delim(inFile2)

ggplot(data2,aes(x = WNT3A_rep1, y =
WNT3A_rep2)) +
  geom_point()
ggsave('scatter1.png')
```



In-Class Exercise:
Try It Out!

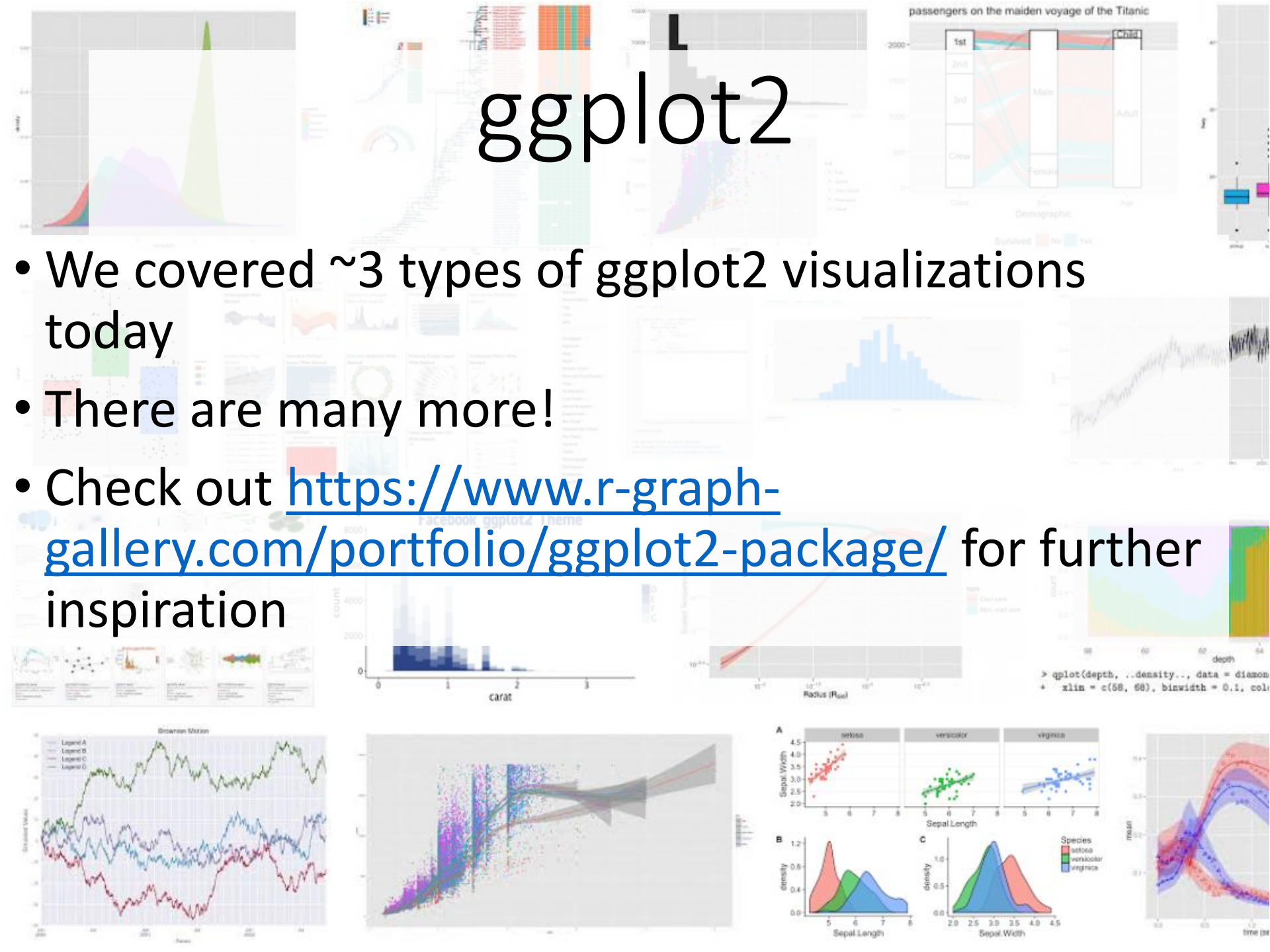
Exercise: Create a boxplot using the code from the previous page and update your visualization to:

1. Change the color and shape of scatter plot points
   - HINT: color = 'blue'
   - HINT: shape = 1
2. Add a black linear regression line using a geom_smooth layer
   - HINT: geom_smooth(method = lm)



ggplot2

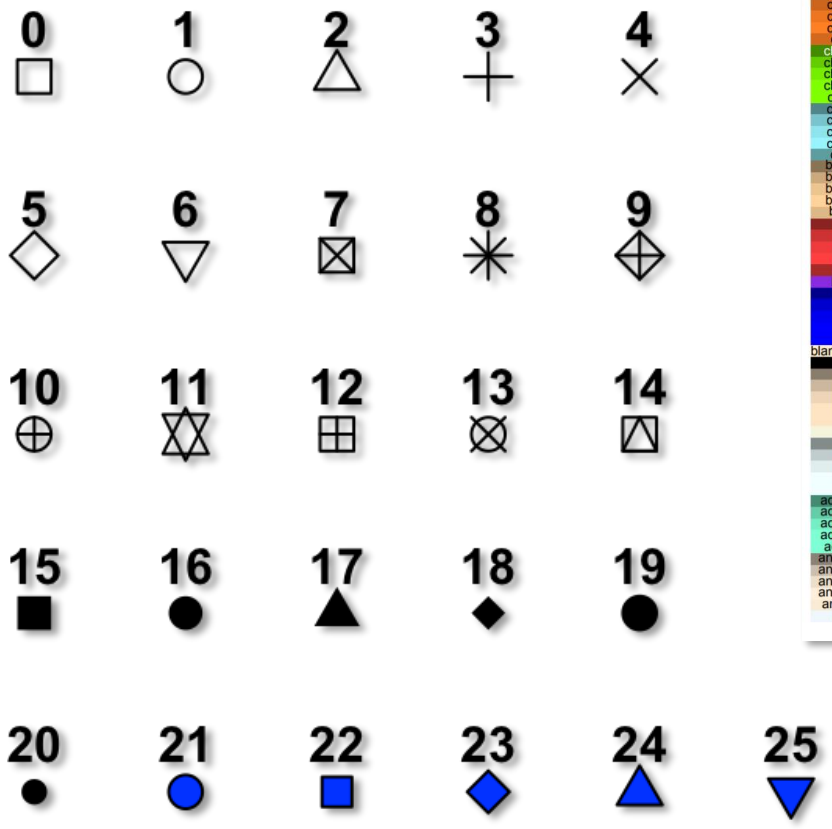
- We covered ~3 types of ggplot2 visualizations today
- There are many more!
- Check out https://www.r-graph-gallery.com/portfolio/ggplot2-package/ for further inspiration



References

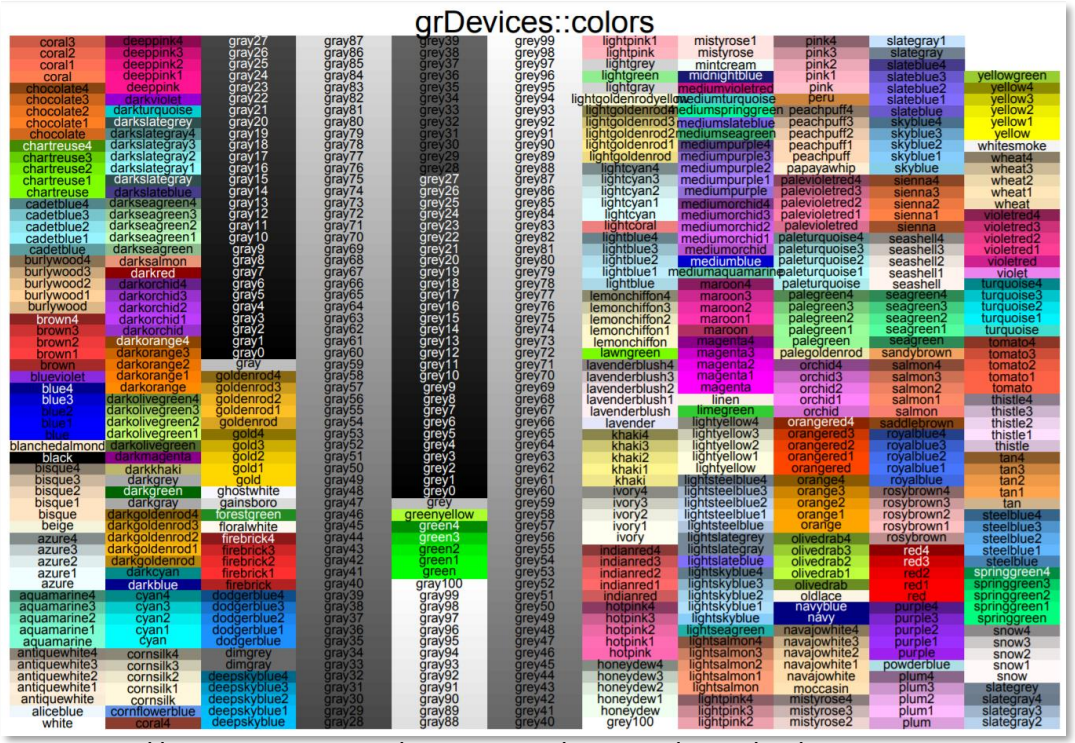
- Gentleman, Robert. R Programming for Bioinformatics. CRC Press, 2009.
- Slides sourced in part from Jacob Kitzman and Barry Grant

R Graphics Shapes



http://www.sthda.com/english/wiki/r-plot-pch-symbols-the-different-point-shapes-available-in-r

R Graphics Colors



http://bc.bojanorama.pl/wp-content/uploads/2013/04/rcolorsheet.pdf

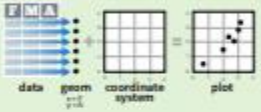
Data Visualization with ggplot2

Cheat Sheet

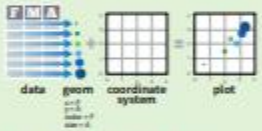
RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

```
qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
```
(aesthetic mappings, data, geom)

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method ="lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```
(data; add layers, elements with +; layer = geom + default stat + layer specific mappings; additional elements)

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()

Returns the last plot

ggsave("plot.png", width = 5, height = 5)

Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

```
a <- ggplot(mpg, aes(hwy))
```

a + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + geom_dotplot()
x, y, alpha, color, fill

a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + geom_histogram(binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

```
b <- ggplot(mpg, aes(fl))
```

b + geom_bar()
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

```
c <- ggplot(map, aes(long, lat))
```

c + geom_polygon(aes(group = group)**)**
x, y, alpha, color, fill, linetype, size

```
d <- ggplot(economics, aes(date, unemploy))
```

d + geom_path(lineend="butt", linejoin="round", linemitre=1**)**
x, y, alpha, color, linetype, size

d + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)**)**
x, ymax, ymin, alpha, color, fill, linetype, size

```
e <- ggplot(seals, aes(x = long, y = lat))
```

e + geom_segment(aes(
xend = long + delta_long,
yend = lat + delta_lat)**)**
x, xend, y, yend, alpha, color, linetype, size

e + geom_rect(aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

```
f <- ggplot(mpg, aes(cty, hwy))
```

f + geom_blank()

f + geom_jitter()
x, y, alpha, color, fill, shape, size

f + geom_point()
x, y, alpha, color, fill, shape, size

f + geom_quantile()
x, y, alpha, color, linetype, size, weight

f + geom_rug(sides = "bl"**)**
alpha, color, linetype, size

f + geom_smooth(model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + geom_text(aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

```
g <- ggplot(mpg, aes(class, hwy))
```

g + geom_bar(stat = "identity"**)**
x, y, alpha, color, fill, linetype, size, weight

g + geom_boxplot()
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + geom_dotplot(binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

g + geom_violin(scale = "area"**)**
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

```
h <- ggplot(diamonds, aes(cut, color))
```

h + geom_jitter()
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

```
i <- ggplot(movies, aes(year, rating))
```

i + geom_bin2d(binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + geom_density2d()
x, y, alpha, colour, linetype, size

i + geom_hex()
x, y, alpha, colour, fill size

Continuous Function

```
j <- ggplot(economics, aes(date, unemploy))
```

j + geom_area()
x, y, alpha, color, fill, linetype, size

j + geom_line()
x, y, alpha, color, linetype, size

j + geom_step(direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

k + geom_crossbar(fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + geom_errorbar()
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + geom_linerange()
x, ymin, ymax, alpha, color, linetype, size

k + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

```
data <- data.frame(murder = USArrests$Murder,
  state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))
```

l + geom_map(aes(map_id = state), map = map**) +**
expand_limits(x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Three Variables

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))
```

m + geom_contour(aes(z = z)**)**
x, y, z, alpha, colour, linetype, size, weight

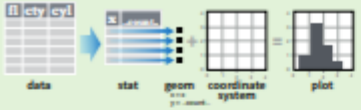
m + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

m + geom_tile(aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size

https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Stats - An alternative way to build a layer

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. **a + geom_bar(stat = "bin")**



Each stat creates additional variables to map aesthetics to. These variables use a common **..name..** syntax.

stat functions and geom functions both combine a stat with a geom to make a layer, i.e. **stat_bin(geom="bar")** does the same as **geom_bar(stat="bin")**

stat function | layer specific mappings | variable created by transformation

i + stat_density2d(aes(fill = ..level..), geom = "polygon", n = 100)

geom for layer | parameters for stat

1D distributions

a + stat_bin(binwidth = 1, origin = 10**)**
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + stat_bindot(binwidth = 1, binaxis = "x"**)**
x, y, | ..count.., ..ncount..
a + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y, | ..count.., ..density.., ..scaled..

2D distributions

f + stat_bin2d(bins = 30, drop = TRUE**)**
x, y, fill | ..count.., ..density..
f + stat_binhex(bins = 30**)**
x, y, fill | ..count.., ..density..
f + stat_density2d(contour = TRUE, n = 100**)**
x, y, color, size | ..level..

3 Variables

m + stat_contour(aes(z = z)**)**
x, y, z, order | ..level..
m+ stat_spoke(aes(radius= z, angle = z)**)**
angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + stat_summary_hex(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..
m + stat_summary2d(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..

Comparisons

g + stat_boxplot(coef = 1.5**)**
x, y | ..lower.., ..middle.., ..upper.., ..outliers..
g + stat_ydensity(adjust = 1, kernel = "gaussian", scale = "area"**)**
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

Functions

f + stat_ecdf(n = 40**)**
x, y | ..x.., ..y..
f + stat_quantile(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq"**)**
x, y | ..quantile.., ..x.., ..y..
f + stat_smooth(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95**)**
x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

General Purpose

ggplot() + stat_function(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5)**)**
x | ..y..
f + stat_identity()
ggplot() + stat_qq(aes(sample=1:100), distribution = qt, dparams = list(df=5)**)**
sample, x, y | ..x.., ..y..
f + stat_sum()
x, y, size | ..size..
f + stat_summary(fun.data = "mean_cl_boot"**)**
f + stat_unique()

Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

n <- b + geom_bar(aes(fill = fl)**)**
n

n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

scale_ | aesthetic to adjust | prepackaged scale to use | scale specific arguments

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

General Purpose scales

Use with any aesthetic:
alpha, color, fill, linetype, shape, size

scale_*_continuous() - map cont' values to visual values
scale_*_discrete() - map discrete values to visual values
scale_*_identity() - use data values **as** visual values
scale_*_manual(values = c()**)** - map discrete values to manually chosen visual values

X and Y location scales

Use with x or y aesthetics (x shown here)

scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks")**)** - treat x values as dates. See ?strptime for label formats.
scale_x_datetime() - treat x values as date times. Use same arguments as scale_x_date().
scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

Color and fill scales

Discrete

n <- b + geom_bar(aes(fill = fl)**)**

n + scale_fill_brewer(palette = "Blues"**)**
For palette choices:
library(RColorBrewer)
display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red"**)**

Continuous

o <- a + geom_dotplot(aes(fill = ..x..)**)**

o + scale_fill_gradient(low = "red", high = "yellow"**)**

o + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25**)**

o + scale_fill_gradientn(colours = terrain.colors(6)**)**
Also: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

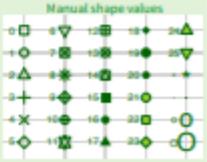
Shape scales

p <- f + geom_point(aes(shape = fl)**)**

p + scale_shape(solid = FALSE**)**

p + scale_shape_manual(values = c(3:7)**)**
Shape values shown in chart on right

Manual shape values



Size scales

q <- f + geom_point(aes(size = cyl)**)**

q + scale_size_area(max = 6**)**
Value mapped to area of circle (not radius)

Coordinate Systems

r <- b + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)**
xlim, ylim
The default cartesian coordinate system

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

r + coord_polar(theta = "x", direction=1 **)**
theta, start, direction
Polar coordinates

r + coord_trans(xtrans = "sqrt"**)**
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set extras and strains to the name of a window function.

z + coord_map(projection = "ortho", orientation=c(41, -74, 0)**)**
projection, orientation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(. ~ fl)
facet into columns based on fl
t + facet_grid(year ~ .)
facet into rows based on year
t + facet_grid(year ~ fl)
facet into both rows and columns
t + facet_wrap(~ fl)
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets
t + facet_grid(y ~ x, scales = "free")
x and y axis limits adjust to individual facets
- **"free_x"** - x axis limits adjust
- **"free_y"** - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(. ~ fl, labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(. ~ fl, labeller = label_bquote(alpha ^ .(x)**))**

α^c	α^d	α^e	α^p	α^r

t + facet_grid(. ~ fl, labeller = label_parsed)

c	d	e	p	r

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(position = "dodge")
Arrange elements side by side

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height

s + geom_bar(position = "stack")
Stack elements on top of one another

f + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting

Each position adjustment can be recast as a function with manual **width** and **height** arguments

s + geom_bar(position = position_dodge(width = 1))

Labels

t + ggtitle("New Plot Title")
Add a main title above the plot
t + xlab("New X label")
Change the label on the X axis
t + ylab("New Y label")
Change the label on the Y axis
t + labs(title =" New title", x = "New x", y = "New y"**)**
All of the above

Use scale functions to update legend labels

Legends

t + theme(legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or "right"
t + guides(color = "none"**)**
Set legend type for each aesthetic: colorbar, legend, or none (no legend)
t + scale_fill_discrete(name = "Title", labels = c("A", "B", "C"**)**
Set legend title and labels with a scale function.

Themes

r + theme_bw()
White background with grid lines
r + theme_grey()
Grey background (default theme)
r + theme_classic()
White background no gridlines
r + theme_minimal()
Minimal theme

ggthemes - Package with additional ggplot2 themes

Zooming

Without clipping (preferred)
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20)**)**

With clipping (removes unseen data points)
t + xlim(0, 100**) + ylim(**10, 20**)**
t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**

https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf