Reporting data results #1

Plot guidelines

Guidelines for good plots

There are a number of very thoughtful books and articles about creating graphics that effectively communicate information.

Some of the authors I highly recommend (and from whose work I've pulled the guidelines for good graphics we'll talk about this week) are:

- Edward Tufte
- Howard Wainer
- Stephen Few
- Nathan Yau

You should plan, in particular, to read *The Visual Display of Quantitative Information* by Edward Tufte before you graduate.

Guidelines for good plots

This week, we'll focus on six guidelines for good graphics, based on the writings of these and other specialists in data display.

The guidelines are:

1. Aim for high data density.
2. Use clear, meaningful labels.
3. Provide useful references.
4. Highlight interesting aspects of the data.
5. Make order meaningful.
6. When possible, use small multiples.

Example data

You can load the data for today's examples with the following code:

```r
library(faraway)
data(nepali)
data(worldcup)

library(dlnm)
data(chicagoNMMAPS)
chic <- chicagoNMMAPS
chic_july <- chic %>%
  filter(month == 7 & year == 1995)
```

High data density

Guideline 1: **Aim for high data density.**

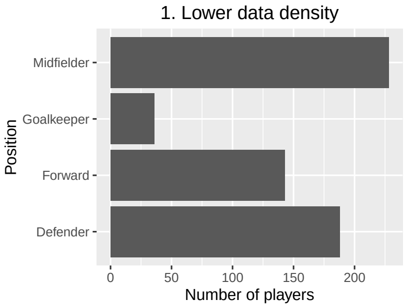
You should try to increase, as much as possible, the **data to ink ratio** in your graphs. This is the ratio of "ink" providing information to all ink used in the figure.

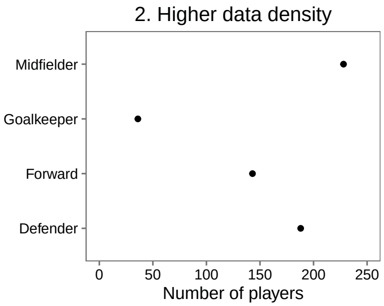
One way to think about this is that the only graphs you make that use up a lot of your printer's ink should be packed with information.

High data density

Guideline 1: **Aim for high data density.**

The two graphs below show the same information. Compare the amount of ink used in the left plot to the amount used in the right plot to see how graphs with the same information can have very different data densities.

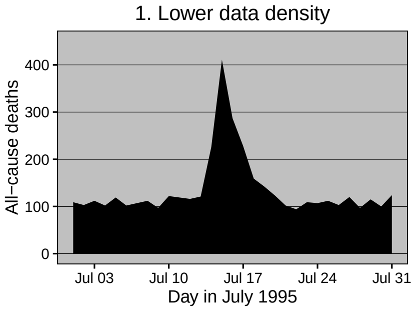


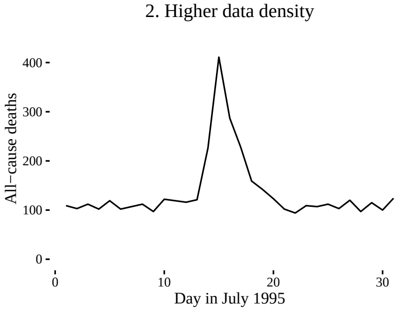


High data density

Guideline 1: **Aim for high data density.**

The two graphs below show another example of very different data densities in two plots showing the same information:





Data density

One quick way to increase data density in `ggplot2` is to change the *theme* for the plot. This essentially changes the "background" elements to a plot, including elements like the plot grid, background color, and the font used for labeling.

Some themes come with `ggplto2`, including:

- `theme_bw`
- `theme_minimal`
- `theme_void`

The `ggthemes` packages has some excellent additional themes.

Data density

The following slides show some examples of the effects of using different themes. The following code creates a plot of daily deaths in Chicago in July 1995:

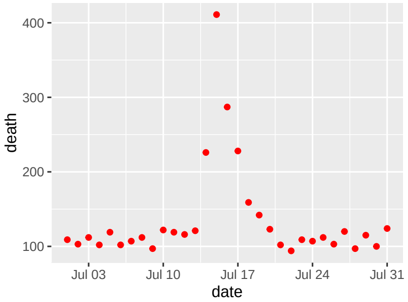
```r
chic_plot <- ggplot(chic_july, aes(x = date, y = death))  +
        geom_point(color = "red")
```

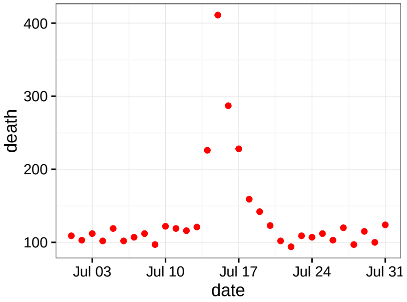
Next, we can see how the graph looks with the default theme and with other themes.

Themes

The left graph shows the graph with the default theme, while the right shows the effect of adding the black-and-white theme that comes with ggplot2 as theme_bw:

```
a <- chic_plot
b <- chic_plot + theme_bw()
grid.arrange(a, b, ncol = 2)
```

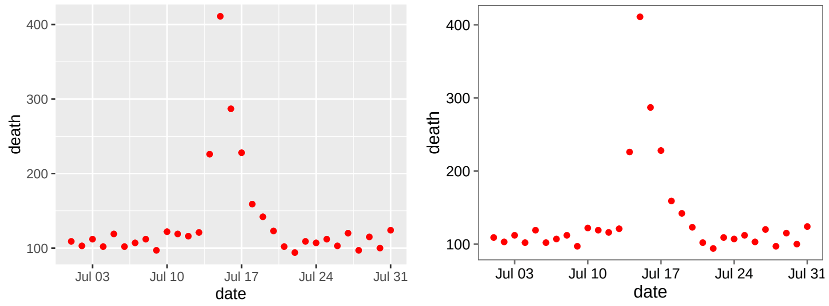




Themes

Stephen Few theme:

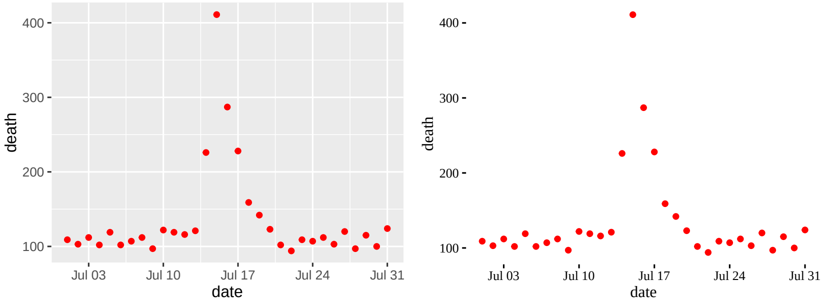
```r
a <- chic_plot
b <- chic_plot + theme_few()
grid.arrange(a, b, ncol = 2)
```



Themes

Edward Tufte theme:

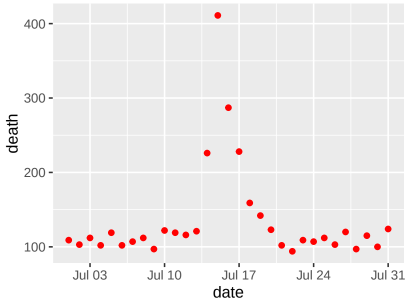
```r
a <- chic_plot
b <- chic_plot + theme_tufte()
grid.arrange(a, b, ncol = 2)
```

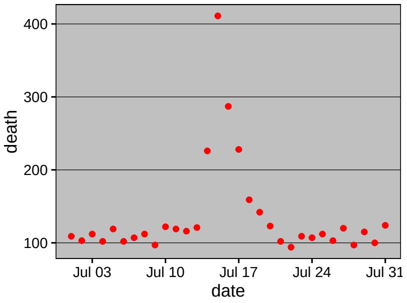


Themes

You can even use themes to add some questionable choices for different elements. For example, ggthemes includes an Excel theme:

```
a <- chic_plot
b <- chic_plot + theme_excel()
grid.arrange(a, b, ncol = 2)
```

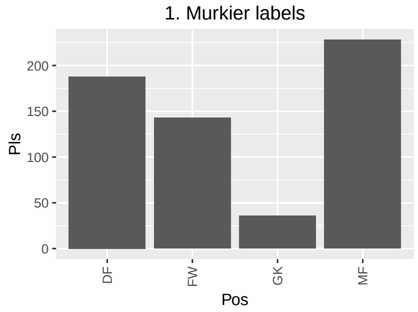


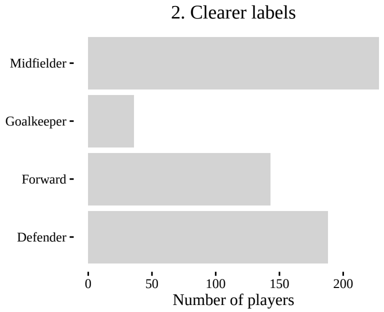


Meaningful labels

Guideline 2: **Use clear, meaningful labels.**

Graph defaults often use abbreviations for axis labels and other labeling. Further, text labels can sometimes be aligned in a way that makes them hard to read. The plots below give an example of the same information shown without (left) and with (right) clear, meaningful labels.





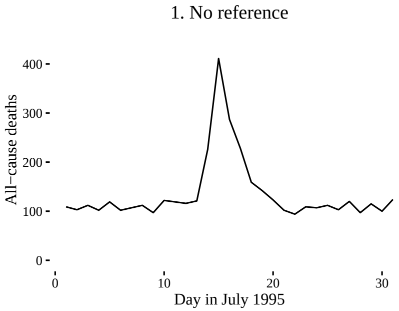
Meaningful labels

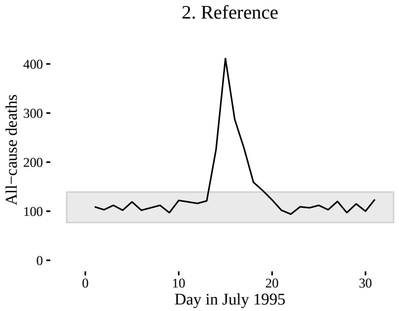
There are a few strategies you can use to make labels clearer:

- Add `xlab` and `ylab` elements to the plot, rather than relying on the column names in the original data. This can also be done with `scale` elements (e.g., `scale_x_continuous`), which give you more power to also make other changes to the x- and y-axes.
- Include units of measurement in parentheses in axis titles when relevant. If units are dollars or percent, check out the `scales` package, which allows you to add labels directly to axis elements by including arguments like `labels = percent` when adding `scale` elements.
- If the x-variable requires longer labels (player positions in the example above), consider flipping the coordinates, rather than abbreviating or rotating the labels. You can use `coord_flip` to do this.

References

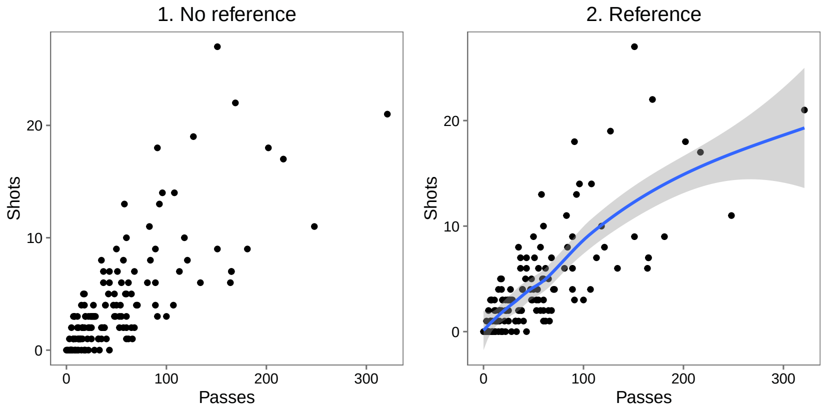
- Guideline 3: **Provide useful references.**





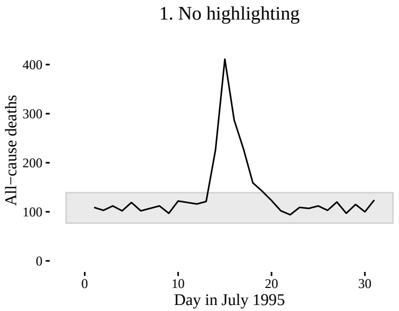
References

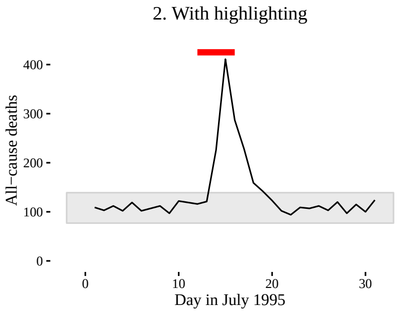
- Guideline 3: **Provide useful references.**



Interesting aspects

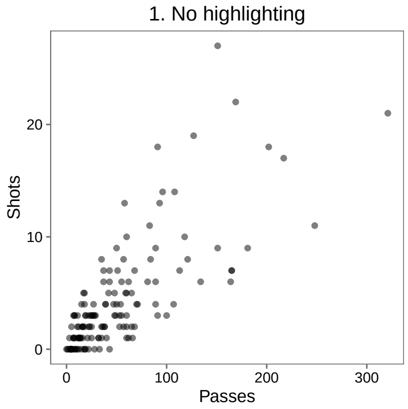
- Guideline 3: **Highlight interesting aspects.**

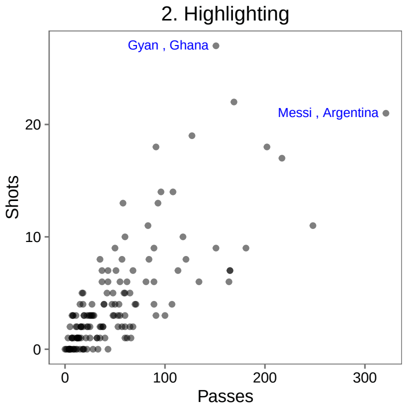




Interesting aspects

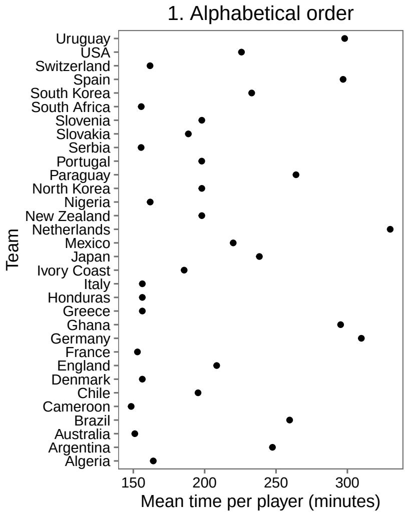
- Guideline 3: **Highlight interesting aspects.**

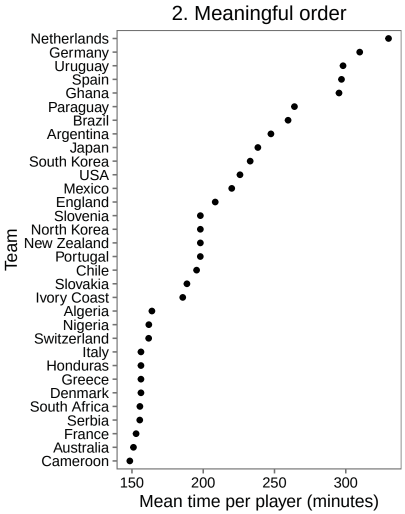




Order

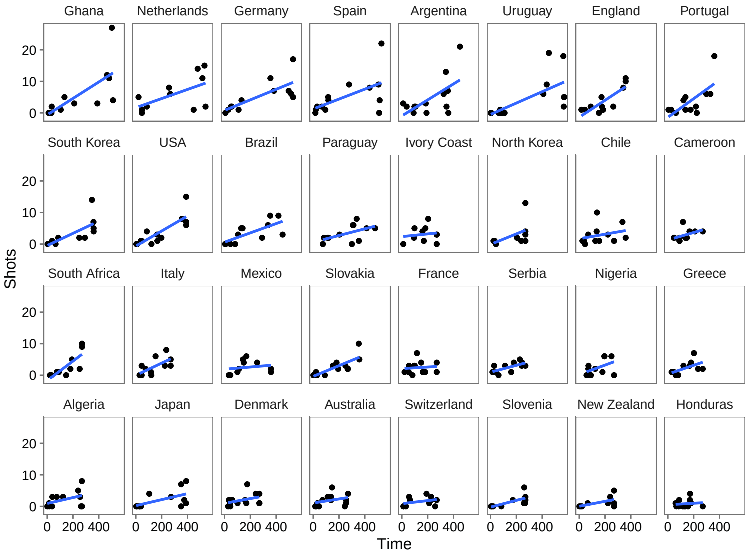
- Guideline 4: **Make order meaningful.**





Small multiples

- Guideline 5: **When possible, use small multiples.**
- Many small plots showing the same thing for different facets
- Same x- and y-axes
- Limit axis annotation


Ghana
Netherlands
Germany
Spain
Argentina
Uruguay
England
Portugal
South Korea
USA
Brazil
Paraguay
Ivory Coast
North Korea
Chile
Cameroon
South Africa
Italy
Mexico
Slovakia
France
Serbia
Nigeria
Greece
Algeria
Japan
Denmark
Australia
Switzerland
Slovenia
New Zealand
Honduras
Shots
0
10
20
Time
0
200
400

Highlighting

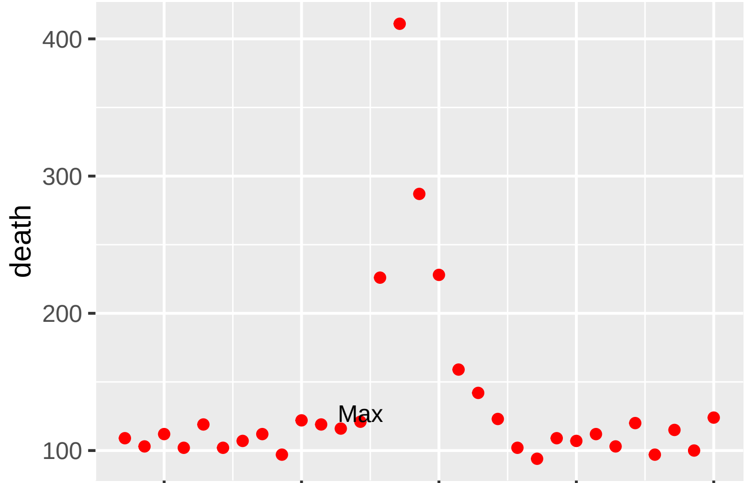
One helpful way to annotate is with text, using `geom_text()`:

```
hottest_day <- chic_july[which.max(chic_july$temp),]
hottest_day
```

```
##                date time year month doy      dow death cvd resp
## 3116 1995-07-13 3116 1995     7 194 Thursday   121  42   10 3
##          dptp  rhum     pm10       o3
## 3116 76.375 57.25 90.36365 57.01744
```

Highlighting

```
chic_plot + geom_text(data = hottest_day,
                      label = "Max",
                      size = 3, hjust = 0.5, vjust = 0)
```



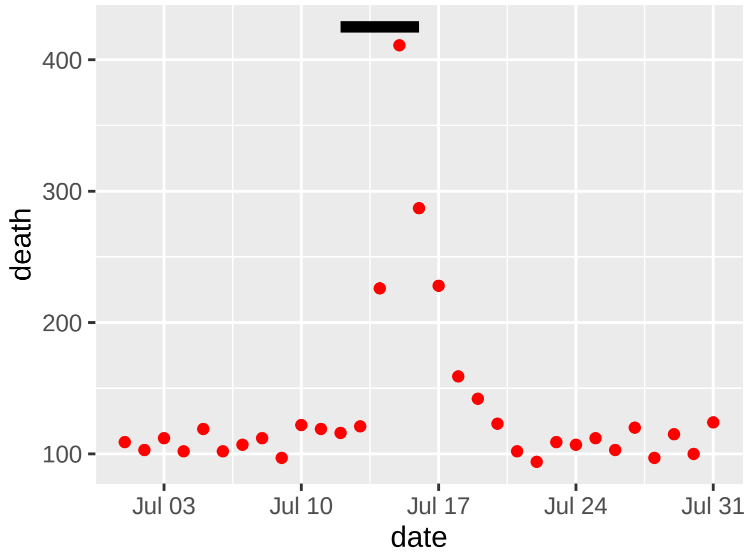
Highlighting

You can also use lines to highlight:

```
chic_july <- subset(chic, month == 7 & year == 1995)
hw <- data.frame(date = c(as.Date("1995-07-12"),
                          as.Date("1995-07-16")),
                 death = c(425, 425))

b <- chic_plot +
        geom_line(aes(x = date, y = death),
                  data = hw[1:2, ],
                  size = 2)
```

b



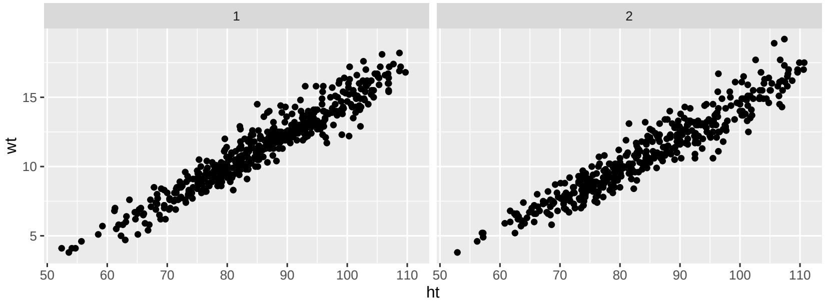
Small multiples

You can use the facet functions to create small multiples. This separates the graph into several small graphs, one for each level of a factor.

- `facet_grid()`
- `facet_wrap()`

Small multiples

```r
ggplot(nepali, aes(ht, wt)) +
        geom_point() +
        facet_grid(. ~ sex)
```



Small multiples

Conventions for `facet_grid`:

```
## Note: This is pseudocode and won't run.
facet_grid([factor for rows] ~ [factor for columns])
```

If you need a different "shape" for faceting, try `facet_wrap()` (in `gridExtra` package):

```
## Note: This is pseudocode and won't run.
facet_wrap(~ [factor for faceting], ncol = [# of columns])
```

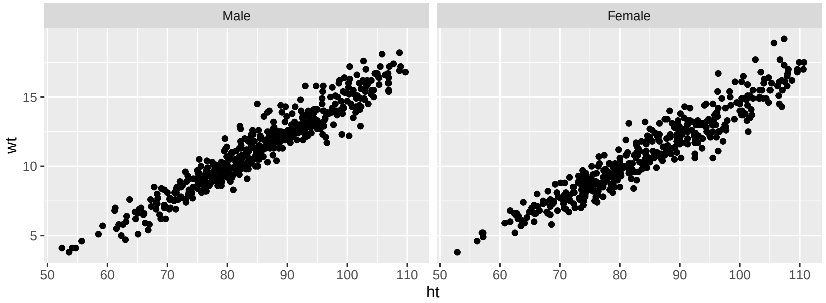
Small multiples

Re-naming and re-ordering factors:
Often, when you do faceting, you'll want to re-name your factors levels or re-order them.
For this, you'll need to use the `factor()` function on the original vector:

```
nepali$sex <- factor(nepali$sex,
                     levels = c(1, 2),
                     labels = c("Male", "Female"))
```

Small multiples

```
ggplot(nepali, aes(ht, wt)) +
         geom_point() +
         facet_grid(. ~ sex)
```



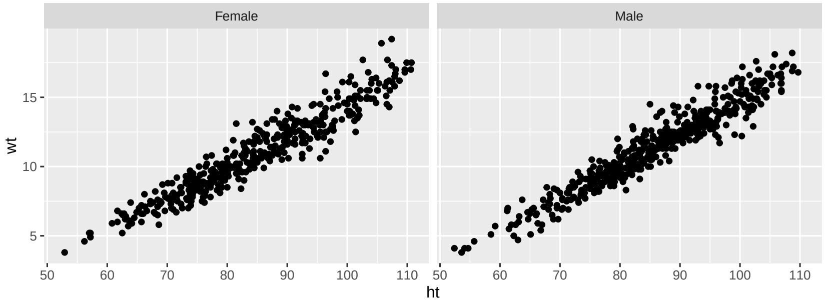
Small multiples

You can do the same idea to re-order a factor:

```
nepali$sex <- factor(nepali$sex,
                     levels = c("Female", "Male"))
```

Small multiples

```
ggplot(nepali, aes(ht, wt)) +
        geom_point() +
        facet_grid(. ~ sex)
```



Annotation

You can use the following functions for main and axis titles:

- `ggtitle()`
- `xlab()`
- `ylab()`

You can also use the “convenience” functions for limits for x- and y-axes:

- `xlim()`
- `ylim()`

EXCELLENT REFERENCES

- R Graphics Cookbook
- Google images

Odds and ends

R cheat sheets

- General (CRAN)
- ggplot
- R Markdown (next week)
- Data wrangling (two weeks from now)
- Data management
- quandl

PICKING PACKAGES

How can I find a reliable package?

- Google “r [what I’m trying to do] tutorial pdf”. Look for tutorials / lecture notes from courses at reputable universities
- Find packages with an accompanying article in the *Journal of Statistical Software* or *R Journal* (or another peer-reviewed publication)
- Use packages recommended by statistics textbooks from our library
- Look for packages referenced in peer-reviewed research in my field
- For a package for graphics, if it works, it works. For methods, be a bit more cautious.