Basic statistical computing in R

Malachy Campbell

11/9/2018

Outline

- Reshaping data
  - Manipulating datasets with tidyr
  - Applying functions over datasets (for loops, apply and plyr)
- Data visualization
  - Base graphics and ggplot2
- Presenting data
  - Reproducible reports and presentations in Rmarkdown

Reshaping data

- We often want to go between ‘long’ and ‘wide’ formats
- Long:

```
##    Line Rep Location     Value
## 1 Line1   1     Home  46.57751
## 2 Line1   2     Home  71.13049
## 3 Line1   3     Home  41.15104
## 4 Line1   1     Away 113.06042
## 5 Line1   2     Away 104.80377
## 6 Line1   3     Away 110.50241
```

- Wide:

```
##    Line Rep      Away     Home
## 1 Line1   1 113.06042 46.57751
## 2 Line1   2 104.80377 71.13049
## 3 Line1   3 110.50241 41.15104
## 4 Line2   1  94.91813 45.99226
## 5 Line2   2  88.39580 46.69039
## 6 Line2   3 106.61620 53.15883
```

Reshaping data: ‘spread’ and ‘gather’

- Rice data set (Zhao et al 2011): 34 traits for 413 rice accessions (not all have phenotypes)
  - ‘Wide’ format: traits listed as different columns
  - We’ll use flowering time at three locations as an example

```
Rice <- read.table(url("http://ricediversity.org/data/sets/44kgwas/RiceDiversity_44K_Phenotypes_34traits_P
                       sep = "\t", header = T)[1:5]
dim(Rice)
```

```
## [1] 413   5
```

Reshaping data: ‘spread’ and ‘gather’ functions

- Convert it to long format using the gather function
  - gather(data, name for combined column, name for value column, names of columns to be combined)

```
Rice_long <- gather(Rice, Location, Value,
                    Flowering.time.at.Arkansas:Flowering.time.at.Aberdeen,
                    factor_key = T)
dim(Rice_long)
```

```
## [1] 1239    4
```

```
dim(Rice)
```

```
## [1] 413   5
```

```
#Give the locations a shorter name
Rice_long$Location <- sub("Flowering.time.at.", "", Rice_long$Location)
```

Reshaping data: ‘spread’ and ‘gather’ functions

- Convert long to wide format using the spread function
  - spread(data, name of the column to be expanded, name for value column)

```r
Rice_wide <- spread(data = Rice_long, key = Location, value = Value)
dim(Rice_wide)
```

```
## [1] 413   5
```

```r
dim(Rice)
```

```
## [1] 413   5
```

Applying functions over data

- In the simplest case we want to get a summary for each trait (flowering time at each location) or for each line
  - If the data is in wide format just take the column means
- Mean flowering time at each location:

```
#Mean of each trait
#colMeans
colMeans(Rice_wide[3:5], na.rm = T)
```

```
##  Aberdeen  Arkansas  Faridpur
## 107.05014  87.94439  71.77049
```

```
#Alternatively use apply and apply the function
#over columns (indicated by 2)
apply(Rice_wide[3:5], 2, mean, na.rm = T)
```

```
##  Aberdeen  Arkansas  Faridpur
## 107.05014  87.94439  71.77049
```

Applying functions over data

- Mean flowering time for each line:

```r
#Mean for each line
head(rowMeans(Rice_wide[3:5], na.rm = T))
```

```
## [1] 101.13889  90.72222  89.12500  89.91667  72.50000  92.08333
```

```r
#Alternatively use apply and apply the function
#over rows (indicated by 1)
head(apply(Rice_wide[3:5], 1, mean, na.rm = T))
```

```
## [1] 101.13889  90.72222  89.12500  89.91667  72.50000  92.08333
```

Applying functions over data

- Suppose we measure a trait at multiple time points and the day of measurement is stored in one column. What can we do if we want to take to take the mean at each time point for each line?

```
Longit <- read.csv("PSA.cleaned.csv", header = T)
head(Longit)
```

```
##   NSFTV.ID Exp Rep DayOfImaging   PSA
## 1  NSFTV_1  E1   1            1 18780
## 2  NSFTV_1  E1   1            2 25434
## 3  NSFTV_1  E1   1            3 29431
## 4  NSFTV_1  E1   1            4 35704
## 5  NSFTV_1  E1   1            5 50058
## 6  NSFTV_1  E1   1            6 59267
```

```
Longit$DayOfImaging <- as.factor(Longit$DayOfImaging)
```

Applying functions over data: for loops

- For each day, subset the data frame, transform it to the wide format and take the mean

```
ResList <- list()
for (i in 1:length(unique(Longit$DayOfImaging))){
  #subset
  tmpdata <- Longit[Longit$DayOfImaging
                    %in% unique(Longit$DayOfImaging)[i] ,]
  tmpdata <- spread(tmpdata, key = NSFTV.ID, value = PSA)
  ResList[[i]] <- colMeans(tmpdata[4:length(tmpdata)], na.rm = T)
}
head(ResList[[20]])
```

```
##   NSFTV_1  NSFTV_10 NSFTV_101 NSFTV_102 NSFTV_103 NSFTV_104
##  318417.3  260199.0  295836.8  390416.0  229428.0  376634.7
```

Applying functions over data: plyr

- Sometimes for loops are necessary, but try to use an alternative if it exists
- ddply: apply some function to subsets of dataframe and combine the results into a nice dataframe

```r
#.(NSFTV.ID, DayOfImaging) these are the columns that we will subset on
Longit_means <- ddply(Longit, .(NSFTV.ID, DayOfImaging), summarise, MeanPSA = mean(PSA, na.rm = T))

head(Longit_means)
```

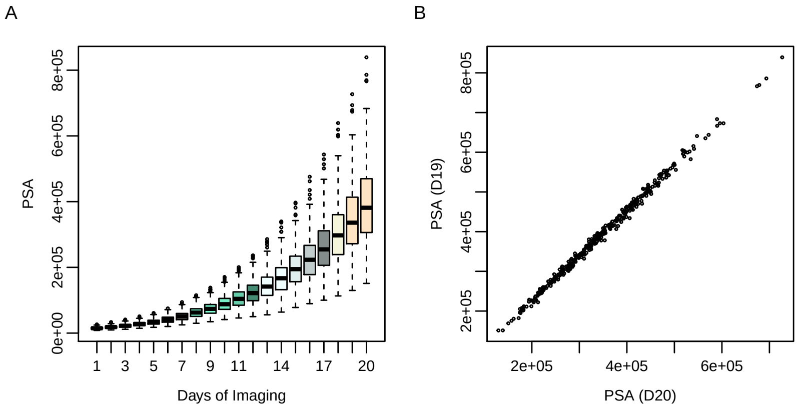
```
##   NSFTV.ID DayOfImaging  MeanPSA
## 1  NSFTV_1            1 16164.00
## 2  NSFTV_1            2 20688.67
## 3  NSFTV_1            3 23542.33
## 4  NSFTV_1            4 29838.00
## 5  NSFTV_1            5 39872.67
## 6  NSFTV_1            6 46087.33
```

- Also check out other plyr functions: dlply, ldply, etc.

Plotting with the R base graphics

```r
pdf("Base_plot.pdf",  h=3, w=6, useDingbats = F,
    pointsize = 10)
par(mar=c(3,4,2,0.5), mgp=c(1.8,0.5,0),
    mfrow=c(1,2), cex.lab = 0.75, cex.axis = 0.75)
#Formula, data, color, etc
boxplot(MeanPSA ~ DayOfImaging, data = Longit_means,
        col = colors()[1:20], ylab = "PSA",
        xlab = "Days of Imaging", cex = 0.3)

mtext("A", line = 1, side = 3, adj = -0.25)

tmp.df <- spread(Longit_means[Longit_means$DayOfImaging %in% c(19,20) ,],
                 DayOfImaging, MeanPSA)

plot(tmp.df$`19`, tmp.df$`20`, pch = 21, cex = 0.3,
     ylab = "PSA (D19)", xlab = "PSA (D20)")
mtext("B", line = 1, side = 3, adj = -0.25)
dev.off()
```

Plotting with the R base graphics

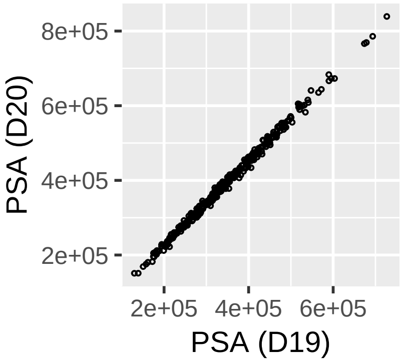


Plotting with ggplot

- Aesthetics: specify the variables in data sets
  - Position, color, shape of points; height of a bar
- Geoms: Specify the type of graph
  - Scatter plot (geom_point), bar (geom_bar), line (geom_line), heatmap (geom_tile)
- Others: statistics, themes, legend, labels, etc.

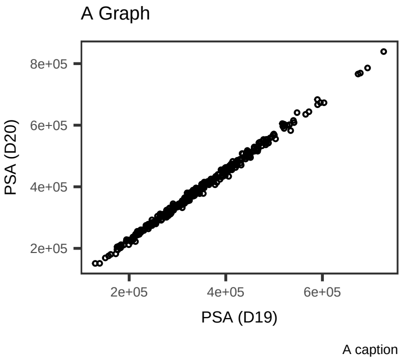
Plotting with ggplot

```
tmp.df <- spread(Longit_means[Longit_means$DayOfImaging %in%
                                c(19,20) ,],
                 DayOfImaging, MeanPSA)

NicePlot <- ggplot(tmp.df) +
  geom_point(aes(x=`19`, y=`20`), pch = 21, size = 0.5) +
  ylab("PSA (D20)") +
  xlab("PSA (D19)")

NicePlot
```



Plotting with ggplot

```
NicePlot <- NicePlot + theme_bw() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          text = element_text(size = 6)) +
          labs(title = "A Graph",
               caption = "A caption")

NicePlot
```

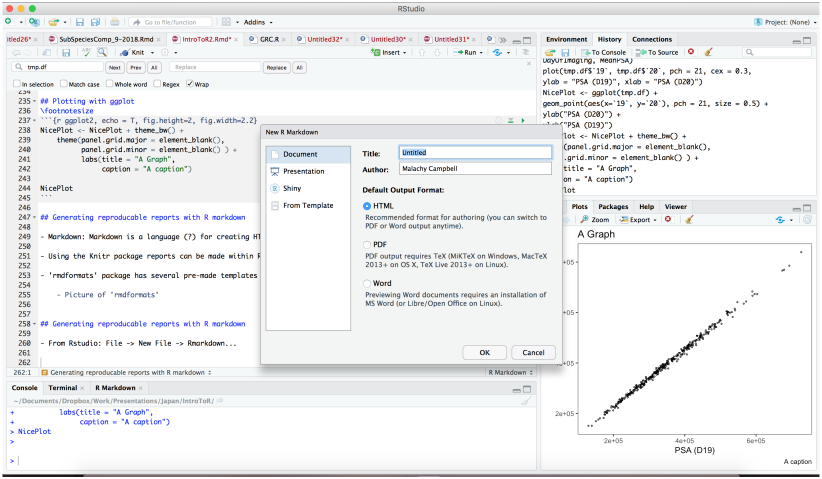


Generating reproducable reports with R markdown

- Markdown: Markdown is a language (?) for creating HTML, PDF, and MS Word
- Using the Knitr package reports can be made within R studio
- ‘rmdformats’ package has several pre-made templates
    - Picture of ‘rmdformats’

Generating reproducable reports with R markdown

- From Rstudio: File -> New File -> Rmarkdown. . .



~~~
## Plotting with ggplot
\footnotesize
```{r ggplot2, echo = T, fig.height=2, fig.width=2.2}
NicePlot <- NicePlot + theme_bw() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank() ) +
          labs(title = "A Graph",
               caption = "A caption")

NicePlot
```


## Generating reproducable reports with R markdown

- Markdown: Markdown is a language (?) for creating HT

- Using the Knitr package reports can be made within R

- 'rmdformats' package has several pre-made templates

    - Picture of 'rmdformats'


## Generating reproducable reports with R markdown

- From Rstudio: File -> New File -> Rmarkdown...
~~~

New R Markdown

- Document
- Presentation
- Shiny
- From Template

Title: Untitled

Author: Malachy Campbell

Default Output Format:

(●) HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).

( ) PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

( ) Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

OK | Cancel

```
DayOfImaging, MeanPSA)
plot(tmp.df$`19`, tmp.df$`20`, pch = 21, cex = 0.3,
ylab = "PSA (D19)", xlab = "PSA (D20)")
NicePlot <- ggplot(tmp.df) +
geom_point(aes(x=`19`, y=`20`), pch = 21, size = 0.5) +
ylab("PSA (D20)") +
  ("PSA (D19)")
lot <- NicePlot + theme_bw() +
(panel.grid.major = element_blank(),
.grid.minor = element_blank() ) +
title = "A Graph",
on = "A caption")
lot
```

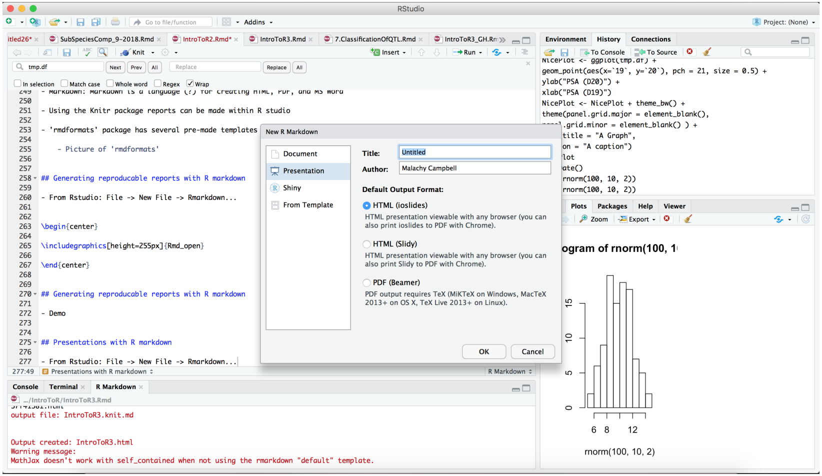
```
~/Documents/Dropbox/Work/Presentations/Japan/IntroToR/
+           labs(title = "A Graph",
+                caption = "A caption")
> NicePlot
>

>
```

Generating reproducable reports with R markdown

- Demo

Presentations with R markdown

- From Rstudio: File -> New File -> Rmarkdown. . .



```
- Markdown: Markdown is a language (?) for creating HTML, PDF, and MS Word

- Using the Knitr package reports can be made within R studio

- 'rmdformats' package has several pre-made templates

    - Picture of 'rmdformats'

## Generating reproducible reports with R markdown

- From Rstudio: File -> New File -> Rmarkdown...


\begin{center}

\includegraphics[height=255px]{Rmd_open}

\end{center}


## Generating reproducible reports with R markdown

- Demo


## Presentations with R markdown

- From Rstudio: File -> New File -> Rmarkdown...
```

New R Markdown

Title: Untitled

Author: Malachy Campbell

Default Output Format:

[✓ HTML (ioslides)] HTML presentation viewable with any browser (you can also print ioslides to PDF with Chrome).

HTML (Slidy) HTML presentation viewable with any browser (you can also print Slidy to PDF with Chrome).

PDF (Beamer) PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

```
NicePlot <- ggplot(tmp.df) +
geom_point(aes(x=`19`, y=`20`), pch = 21, size = 0.5) +
ylab("PSA (D20)") +
xlab("PSA (D19)")
NicePlot <- NicePlot + theme_bw() +
theme(panel.grid.major = element_blank(),
panel.grid.minor = element_blank() ) +
title = "A Graph",
on = "A caption")
lot
ate()
rnorm(100, 10, 2))
rnorm(100, 10, 2))
```

```
output file: IntroToR3.knit.md

Output created: IntroToR3.html
Warning message:
MathJax doesn't work with self_contained when not using the rmarkdown "default" template.
```