Learning *R*

Carl James Schwarz

StatMathComp Consulting by Schwarz
cschwarz.stat.sfu.ca @ gmail.com

Reshaping between wide and long formats for data

Advanced uses.

Table of Contents I

Reshaping Data - Advanced

Wide $\leftrightarrow$ Long formats

Reshaping Data

Wide data format commonly found with many variables or longitudinal data

```
> chick.wide <- read.csv("../sampledata/chickweight.csv",
+                header=TRUE, as.is=TRUE,
+               strip.white=TRUE)
> head(chick.wide)
```

```
> head(chick.wide)
  Chick  Diet Day01 Day02 Day04 Day06 Day08 Day10 Day12 Day1
1     1 Diet1    42    51    59    64    76    93   106   12
2     2 Diet1    40    49    58    72    84   103   122   13
3     3 Diet1    43    39    55    67    84    99   115   13
```

We would like a plot of the mean weight over time for each diet.

Reshaping Data

Long data format transposes each row of data into a long format

```
> head(chick.long)
   Chick Diet  Time Weight
1      1    1 Day01     42
2      1    1 Day02     51
3      1    1 Day04     59
4      1    1 Day06     64
5      1    1 Day08     76
6      1    1 Day10     93
```

Reshaping Data - Why?

- Many statistical models require repeated measure data to be in long format.
- *ggplot()* expects most data to be in long format.
- Quck and dirty way to get plots of multiple variables for screening etc. using faceting in *ggplot()*

Reshaping Data - packages

- Base *R reshape()* function
  - Too hard to use; documentation is useless
- *reshape* - older package do not use
- *reshape2* - deprecated and not updated but still very popular
- *tidyr* - most current but harder to use
- *data.table* - allows for multiple variables to be melted and casted.

Reshaping Data - melting - wide $\rightarrow$ long

tidyr::gather(df,
key="xxx"
value="yyy" c("aaa","bbb","ccc",...))

Separate variables into:

- *key* = variable to hold the names of the transposed variables
- *value* = variable to hold the value of the transposed variables
- ... = variable names to transpose (the measure variables)

All other variables (not in the ...) remain in the wide format.

Reshaping Data - melting - wide $\rightarrow$ long

Example:

```
chick.long <- tidyr::gather(chick.wide,
                    key="Time",          # long descriptor
                    value="Weight",    # long values
                    c("Day01","Day02","Day04",
                      "Day06","Day08","Day10",
                      "Day12","Day14","Day16",
                      "Day18","Day20","Day21")
                   )
head(chick.long)
```

If you leave out the list of measured variables, then all variables will be transposed.

Reshaping Data - melting - wide $\rightarrow$ long

Example:

```
> head(chick.wide)
  Chick  Diet Day01 Day02 Day04 Day06 Day08 Day10 Day12 Day1
1     1 Diet1    42    51    59    64    76    93   106   12

> head(chick.long)
   Chick Diet  Time Weight
1      1    1 Day01     42
2      1    1 Day02     51
3      1    1 Day04     59
4      1    1 Day06     64
5      1    1 Day08     76
6      1    1 Day10     93
```

Reshaping Data - melting - wide → long

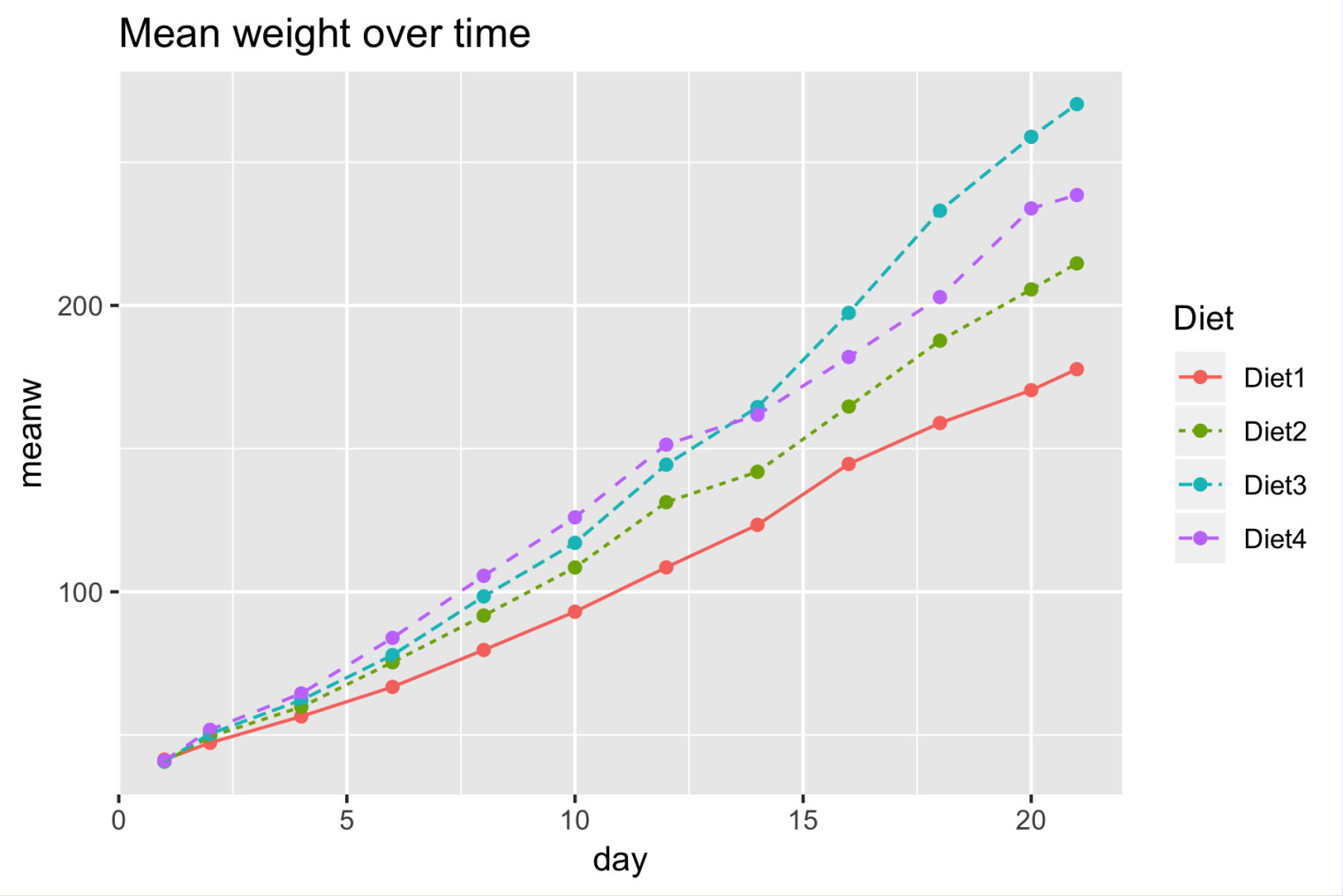
Now we can compute means for each diet x day combination

```
meanw <- plyr::ddply(chick.long, c("Time","Diet"),
                     summarize,
                     day = as.numeric(substring(Time[1],4)),
                     meanw=mean(Weight, na.rm=TRUE))
meanw

plot.meanw <- ggplot2::ggplot(data=meanw,
                              aes(x=day, y=meanw,
                              color=Diet, linetype=Diet))+
                geom_point()+
                geom_line()+
                ggtitle("Mean weight over time")
plot.meanw
```

Plotting with *ggplot2* - Scatterplot

Create a plot of calories vs. grams of fat



Reshaping Data - melting - wide $\rightarrow$ long

Example: melting different variables for plotting purposes: Calories from fat, protein, carbohydrates across shelves.

```
1  head(cereal[,c("name","fat","protein","carbo")])
2
3  cereal$calories.fat     <- cereal$fat * 9
4  cereal$calories.protein <- cereal$protein *4
5  cereal$calories.carbo   <- cereal$carbo * 4
6
7  cereal.long <- tidyr::gather(cereal,
8                      key="Source",
9                      value="Calories",
10                     c("calories.fat",
11                       "calories.protein",
12                       "calories.carbo"))
13 head(cereal.long)
```

Reshaping Data - melting - wide $\rightarrow$ long

Example: comparing calories from different sources.

```
> head(cereal[,c("name","fat","protein","carbo")])
                        name fat protein carbo
1                  100%_Bran   1       4   5.0
2          100%_Natural_Bran   5       3   8.0
3
> head(cereal.long)
                 name shelfF          Source Calories
1           100%_Bran      3     calories.fat        9
2           100%_Bran      3 calories.protein       16
3           100%_Bran      3   calories.carbo       20
4 100%_Natural_Bran      3     calories.fat       45
5 100%_Natural_Bran      3 calories.protein       12
6 100%_Natural_Bran      3   calories.carbo       32
```

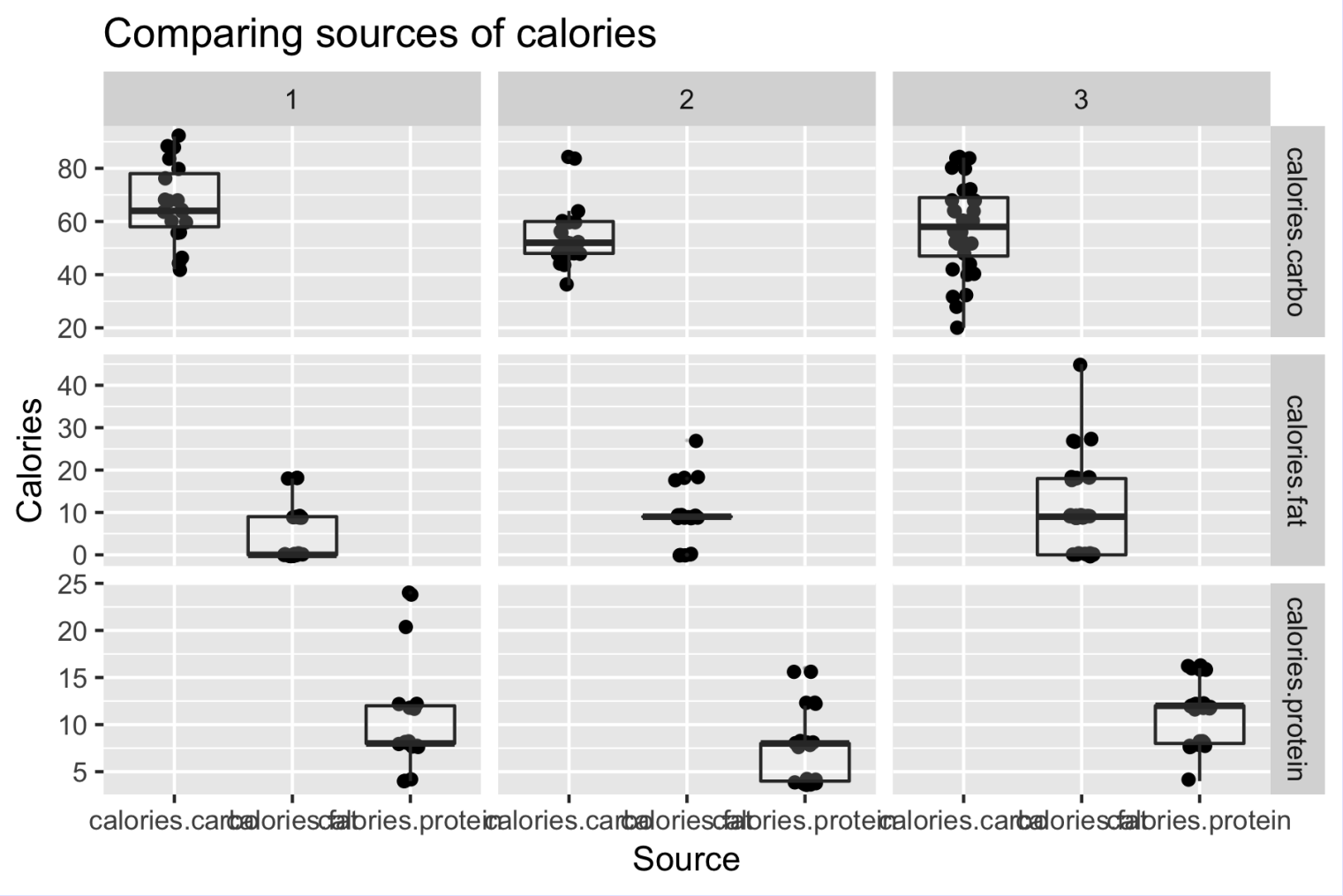
Reshaping Data - melting - wide $\rightarrow$ long

Example: comparing calories from different sources.

```
1 plot1 <- ggplot(data=cereal.long, aes(x=Source, y=Calories))
2    ggtitle("Comparing sources of calories")+
3    geom_point(position=position_jitter(w=0.1))+
4    geom_boxplot(alpha=0.2, outlier.size=0)+
5    facet_grid(Source ~ shelfF, scales="free")
6 plot1
```

Plotting with *ggplot2* - Scatterplot

Create a plot of calories vs. grams of fat



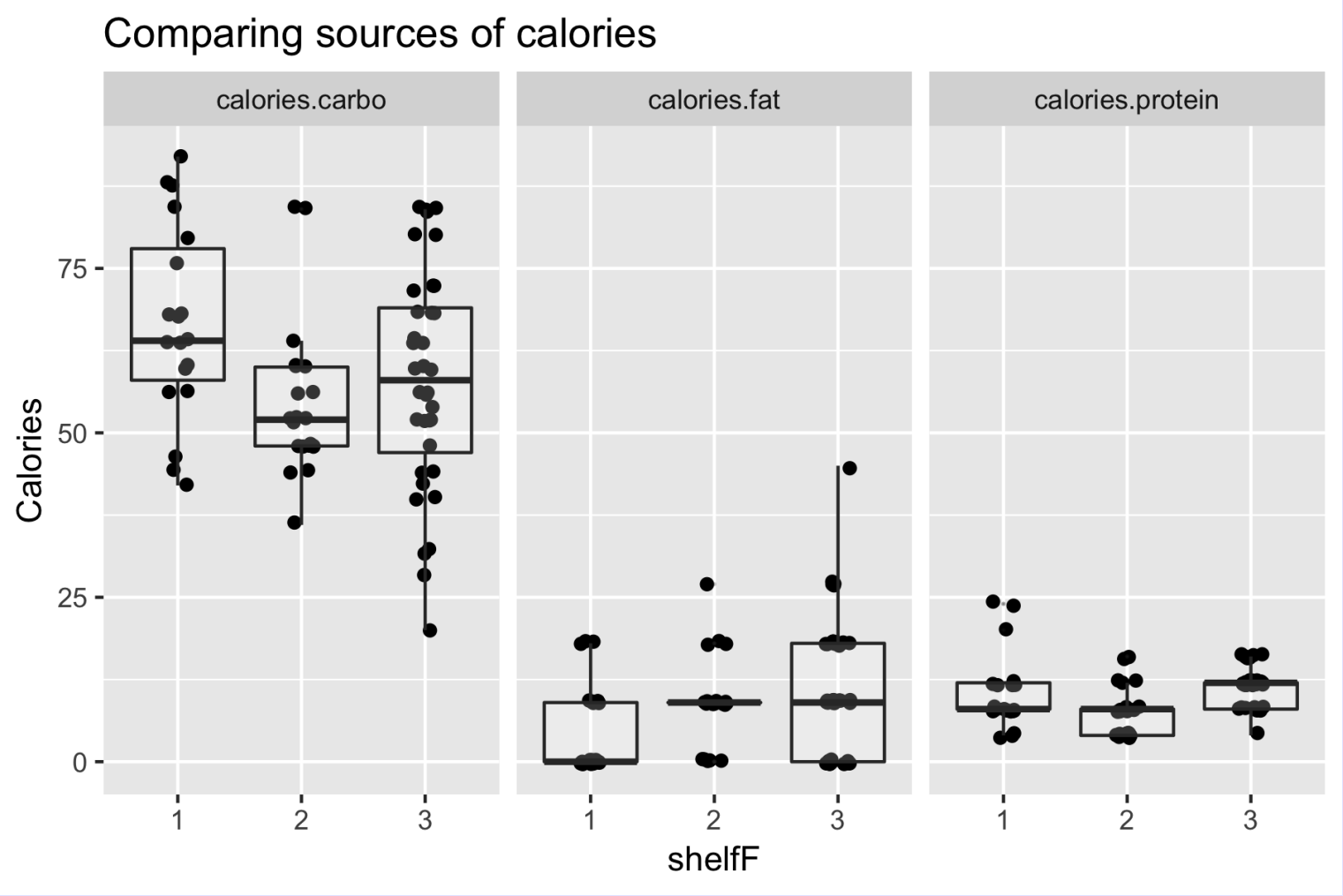
Reshaping Data - melting - wide $\rightarrow$ long

Example: comparing calories from different sources.

```
1 plot2 <- ggplot(data=cereal.long, aes(x=shelfF, y=Calories))
2    ggtitle("Comparing sources of calories")+
3    geom_point(position=position_jitter(w=0.1))+
4    geom_boxplot(alpha=0.2, outlier.size=0)+
5    facet_wrap(~Source )
6 plot2
```

Plotting with *ggplot2* - Scatterplot

Create a plot of calories vs. grams of fat



Reshaping Data - spread - long → wide

Less common:

tidyr::spread(df,
key="xxx",
value="yyyyy")

Example:

```
1 chick.wide2 <- tidyr::spread(chick.long,
2                              key="Time",
3                              value="Weight")
4 head(chick.wide2)
5
```

Reshaping Data - melting - wide $\rightarrow$ long

Example:

```
> head(chick.long)
   Chick Diet  Time Weight
1      1    1 Day01     42
2      1    1 Day02     51
3      1    1 Day04     59
4      1    1 Day06     64
5      1    1 Day08     76
6      1    1 Day10     93

> head(chick.wide2)
  Chick  Diet Day01 Day02 Day04 Day06 Day08 Day10 Day12 Day1
1     1 Diet1    42    51    59    64    76    93   106   12
2     2 Diet1    40    49    58    72    84   103   122   13
```

Reshaping data - Exercise

Teeth dataset - number of types of teeth for mammals

- Read the data
- Sum the top and bottom teeth classification.
- Melt the 4 types of teeth
- Make a nice plot comparing the distribution of teeth by mammal classification (H or C)

Reshaping data - Exercise

```
1  teeth.wide <- read.csv("../sampledata/Teeth.csv",
2                         header=TRUE, strip.white=TRUE,
3                         as.is=TRUE)
4  teeth.wide
5
6  teeth.wide$Incisors <- teeth.wide$Top.incisors + teeth.wide$
7  teeth.wide$Canines  <- teeth.wide$Top.canines  + teeth.wide$
8  teeth.wide$Premolars<- teeth.wide$Top.premolars+ teeth.wide$
9  teeth.wide$Molars   <- teeth.wide$Top.molars   + teeth.wide$
10
11 head(teeth.wide[,c("Mammal","Class","Incisors","Canines","Pr
```

```
> head(exp.long)
> head(teeth.wide[,c("Mammal","Class","Incisors","Canines","
          Mammal Class Incisors Canines Premolars Molars
1         BADGER     c        6       2         6      3
2         COUGAR     c        6       2         5      2
3 ELEPHANT SEAL     c        3       2         8      2
```

Reshaping data - Exercise

```
1  tteeth.long <- tidyr::gather(teeth.wide,
2                               key="Tooth.Type",
3                               value="Teeth",
4                               c("Incisors","Canines","Premolars","Molars
5                               )
6  head(teeth.long)
7
```

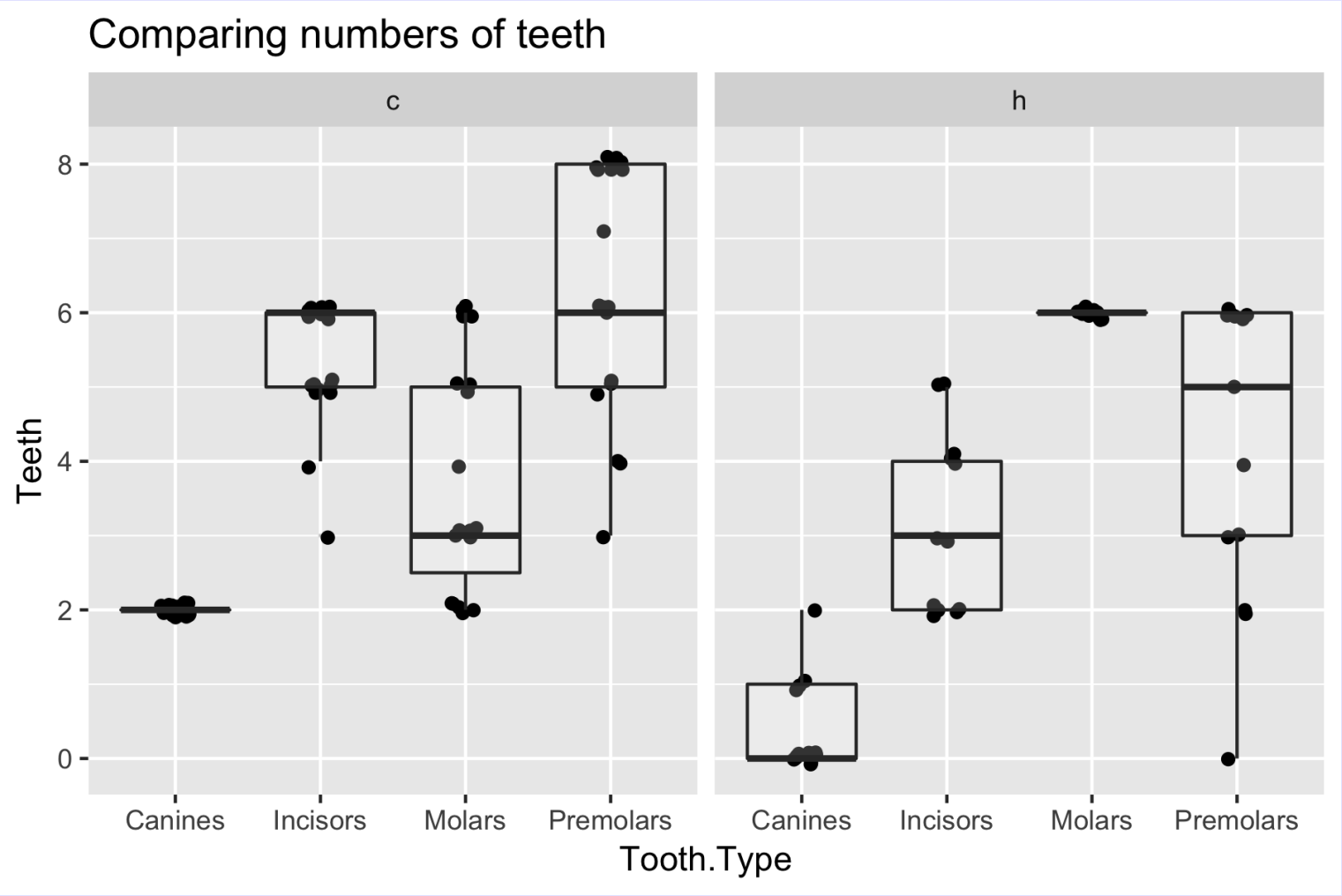
```
> head(teeth.long)
          Mammal Class Tooth.Type Teeth
1         BADGER     c   Incisors     6
2         COUGAR     c   Incisors     6
3 ELEPHANT SEAL     c   Incisors     3
```

Reshaping data - Exercise

```
plot1 <-ggplot(data=teeth.long, aes(x=Tooth.Type, y=Teeth))+
  ggtitle("Comparing numbers of teeth")+
  geom_point(position=position_jitter(h=.1, w=.1))+
  geom_boxplot(alpha=0.2, outlier.size=0)+
  facet_wrap(~Class)
plot1

```

Reshaping data - Exercise



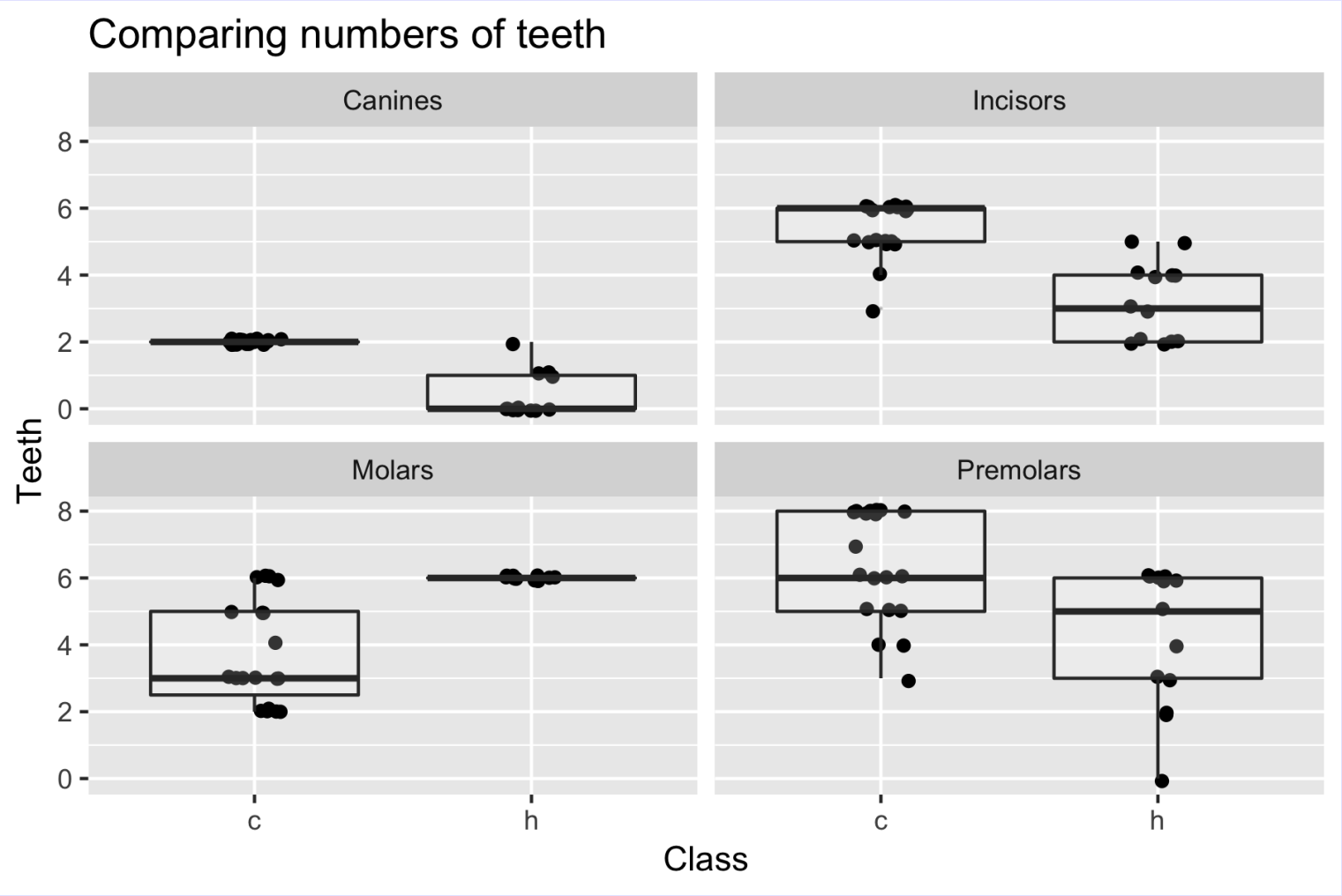
What do you conclude?

Reshaping data - Exercise

```
plot2 <- ggplot(data=teeth.long, aes(x=Class, y=Teeth))+
  ggtitle("Comparing numbers of teeth")+
  geom_point(position=position_jitter(h=.1, w=.1))+
  geom_boxplot(alpha=0.2, outlier.size=0)+
  facet_wrap(~Tooth.Type, ncol=2)
plot2

```

Reshaping data - Exercise



What do you conclude?

Reshaping data - Exercise

Return to the Birds 'n Butts dataset.

Look at the Experimental tab in the Excel file.
Paired design with two halves of 1 nest receiving treatments.

- Read directly in from Excel spreadsheet.
  - How do you skip the first row?
  - How do you fix the variable names?
- Cast to put both values of number of mites on same record
- Compute the difference in the number of mites
- Make a plot to decide if there is an effect?

Reshaping data - Exercise

```
1  exp.long <- readxl::read_excel("../sampledata/bird-butts-dat
2                                 sheet="Experimental", skip=1,
3                                 .name_repair="universal")
4  head(exp.long)
5
```

```
> head(exp.long)
# A tibble: 6 x 5
   Nest Species Nest.content Number.of.mites    Treatment
  <dbl>   <chr>         <chr>           <dbl>        <chr>
1     1    HOSP         empty               0      control
2     1    HOSP         empty               0 experimental
3     2    HOSP         empty               1      control
4     2    HOSP         empty               0 experimental
```

Reshaping data - Exercise

```
1  exp.wide <- tidyr::spread(exp.long,
2                            key="Treatment",
3                            value="Number.of.mites")
4  exp.wide$c.minus.e <- exp.wide$control - exp.wide$experiment
5  head(exp.wide)
6
```

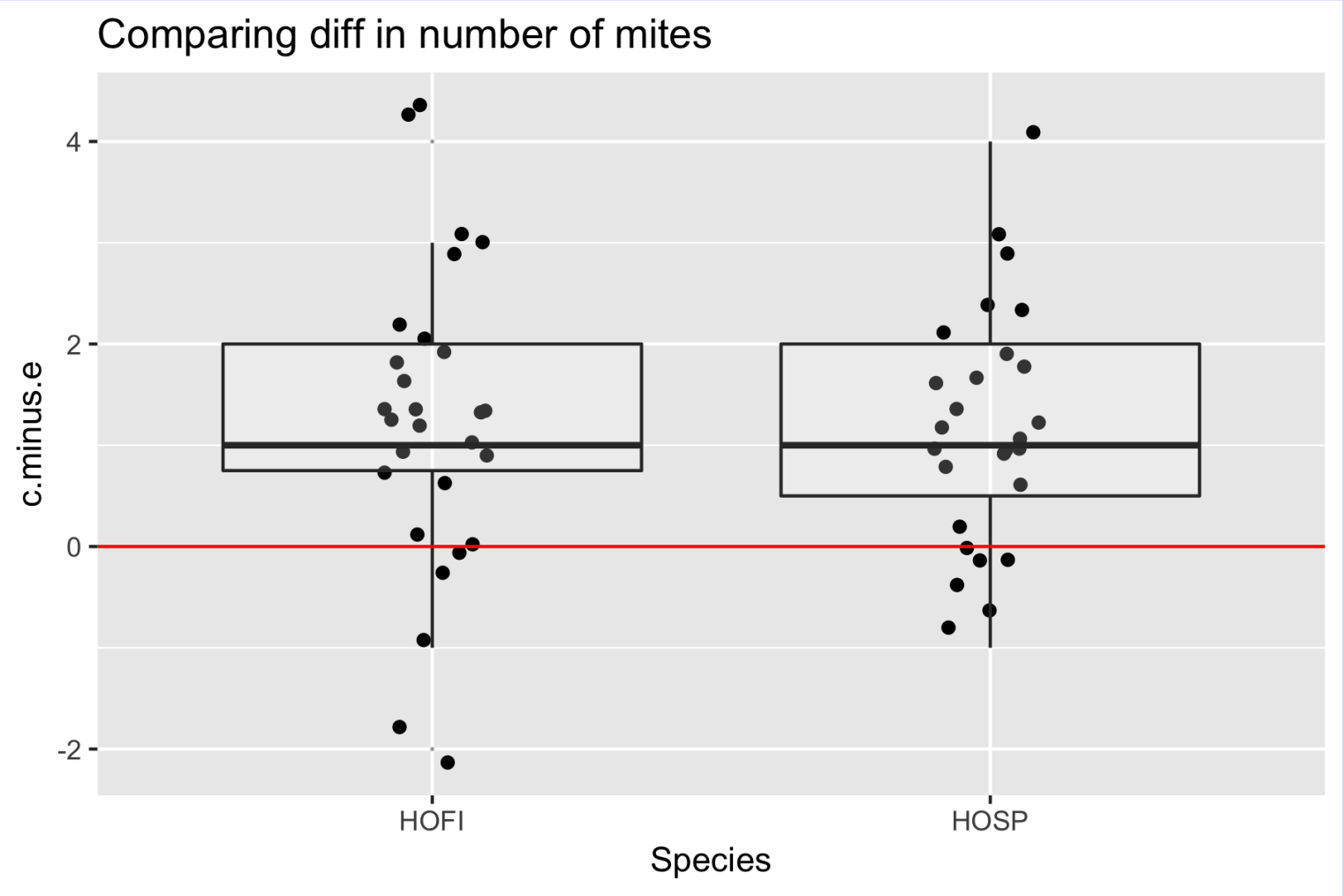
```
> head(exp.wide)
  Nest Species Nest.content control experimental c.minus.e
1    1    HOSP         empty       0            0         0
2    2    HOSP         empty       1            0         1
3    3    HOSP          eggs       3            1         2
```

Reshaping data - Exercise

```
plot1 <- ggplot(data=exp.wide, aes(x=Species, y=c.minus.e))+
  ggtitle("Comparing diff in number of mites")+
  geom_point(position=position_jitter(w=0.1))+
  geom_boxplot(alpha=0.2, outlier.size=0)+
  geom_hline(yintercept=0,color="red")
plot1

```

Reshaping data - Exercise



What do you conclude?

Reshaping Data - Advanced

Wide $\leftrightarrow$ Long formats

Multiple key-value pair.

data.table package.

Reshaping Data - melting - wide → long

Sometime you have multiple key-value pairs.
Refer to the *ncs_teaching_childhealth.xlsx* workbook and the *wide* format worksheet.

```
 CHILD_PIDX VISIT_WT_6 VISIT_WT_12 VISIT_WT_18 VISIT_WT_24 C
  <chr>       <chr>      <chr>       <chr>       <chr>
1 a00058528   18         20          21          NA
2 a00103956   14         NA          24          24
3 a00104038   11         22          25          NA
4 a00104358   18         NA          27          31
5 a00145110   NA         21          24          NA
6 a00145135   21         NA          31          37
# É with 10 more variables: CHILD_AGE_18 <chr>, CHILD_AGE_24
#   GASTRO_12 <chr>, GASTRO_18 <chr>, GASTRO_24 <chr>, EAR_I
#   EAR_INFECTION_12 <chr>, EAR_INFECTION_18 <chr>, EAR_INFE
```

Age, weight, and other variables measured at 4 followup visits and recorded in the wide format.

Reshaping Data - melting - wide $\rightarrow$ long

Example. Very similar to the *reshape2* package.

```
1 library(data.table)
2 child.long <- data.table::melt(as.data.table(child.wide),
3     id.var="CHILD_PIDX",
4     measure.vars=list(c("VISIT_WT_6", "VISIT_WT_12",  "VISI
5                                       c("CHILD_AGE_6","CHILD_AG
6     variable.name="Visit",
7     value.name=c("Weight","Age"))
```

Don't forget to use the *as.data.table()* function otherwise you get an uninformative error message.

Reshaping Data - casting - long → wide

Sometime you have multiple key-value pairs.
Refer to the *ncs_teaching_childhealth.xlsx* workbook and the *long* format worksheet.

```
 > head(child.long)
# A tibble: 6 x 11
  CHILD_PIDX CHILD_AGE VISIT VISIT_WT CHILD_HEALTH GASTRO DI
  <chr>          <dbl> <dbl>    <dbl>        <dbl>  <dbl>
1 a00058528        6.7     6       18            1      2
2 a00058528       12.2    12       20            1      1
3 a00058528       17.4    18       21            1     NA
4 a00103956        5.9     6       14            1      2
5 a00103956       18.6    18       24            1     NA
6 a00103956       23      24       24            1     NA
```

Age, weight, and other variables measured at 4 followup visits and recorded in the long format.

Reshaping Data - casting - long $\rightarrow$ wide

Example. Very similar to the *reshape2* package.

```
1 library(data.table)
2 child.wide <- data.table::dcast(as.data.table(child.long),
3         CHILD_PIDX ~ VISIT,
4         value.var=c("VISIT_WT","CHILD_AGE","GASTRO","EAR_INFE
5 head(child.wide)
```

Don't forget to use the *as.data.table()* function, otherwise you get an uninformative error message.

Reshaping- Summary using *tidyr*

Sometimes necessary to convert from wide $\leftrightarrow$ long formats.
tidyr package.

- Wide to long *tidyr::gather()* is often used prior to making a plot using *ggplot()*.
- Long to wide *tidyr::spread()* is often used to bring elements of paired data together.
- *tidyr* has less functionality than *reshape2* but most of the time you don't need the additional features.

I actually prefer the *reshape2* package.

Reshaping- Summary using *data.table*

Sometimes necessary to convert from wide $\leftrightarrow$ long formats.
data.table package.

- Wide to long *data.table::melt()* is often used prior to making a plot using *ggplot()*.
- Long to wide *data.table::cast()* is often used to bring elements of paired data together.
- *data.table* has more functionality than *reshape2*.
- Don't forget the *as.data.table()* to convert the data.frame to a data.table.