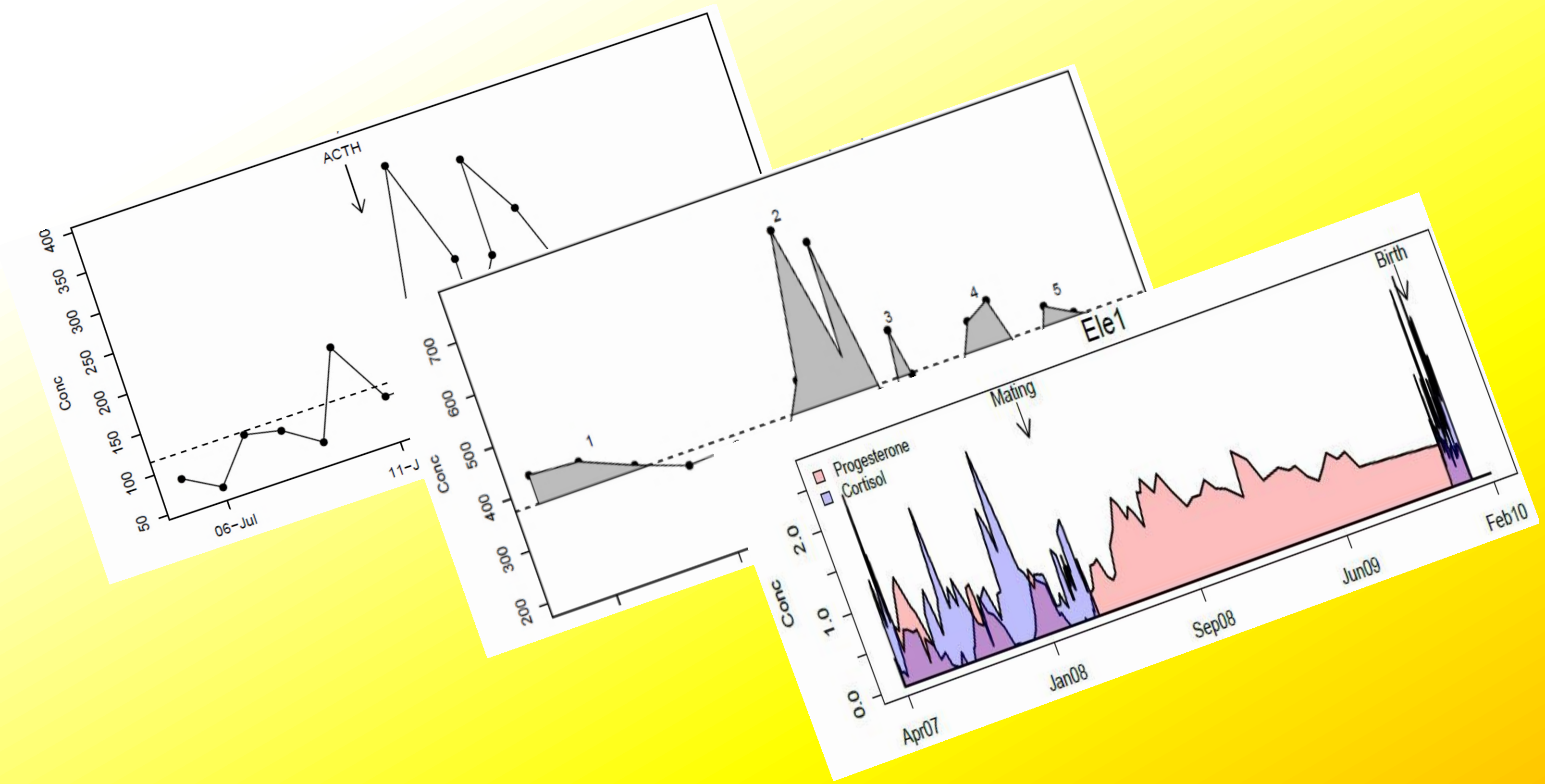
hormLong

An R package for the longitudinal analysis of hormone data

Instruction manual

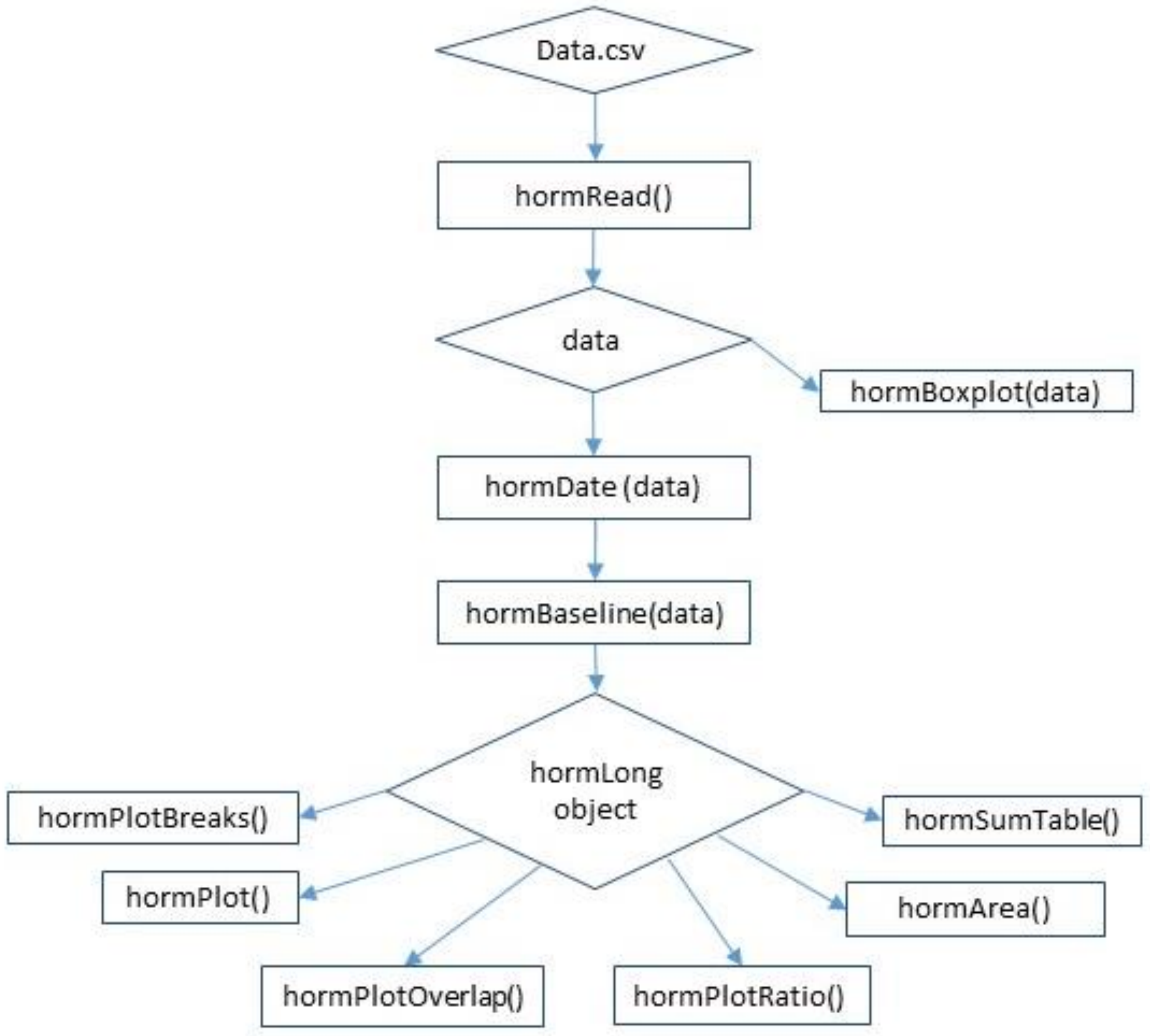


Developed by:
Kerry Fanson and Ben Fanson

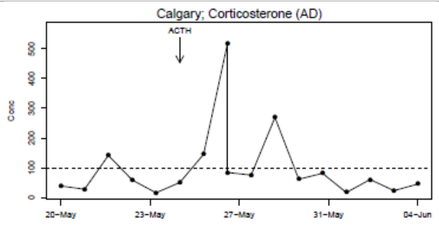
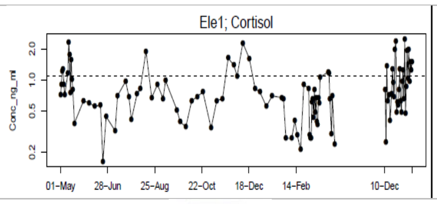
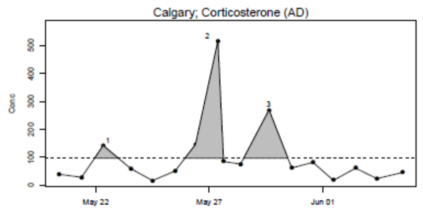
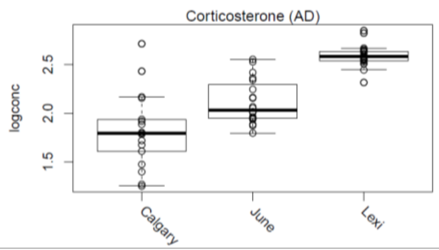
Table of contents

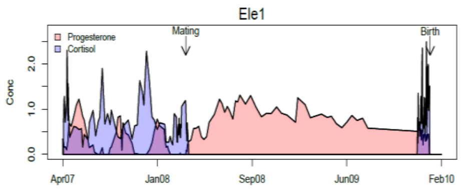
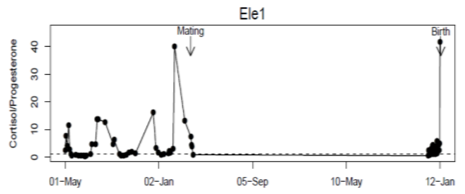
Quick Start Guide

Flowchart



Overview of functions

Function	Purpose	Output
hormRead()	Opens a dialog box to select *csv* file to import	
hormDate()	Fix/standardize date format	
hormBaseline()	Performs iterative baseline calculation on hormone data. Calculations can be grouped using 'by_var' option, allowing you to process multiple individuals/hormones/etc. in 1 step.	*csv* file
hormPlot()	Creates longitudinal plots of hormone data, including reference line for baseline cutoff and events.	
hormPlotBreaks()	Similar to *hormPlot()* but allows you to insert breaks in the x-axis.	
hormSumTable()	Calculates summary statistics for hormone data.	*csv* file
hormArea()	Calculate and plot area under the curve (AUC).	*csv* file 
hormBoxPlot()	Creates boxplots to compare distribution of hormone values for different individuals/populations/etc.	

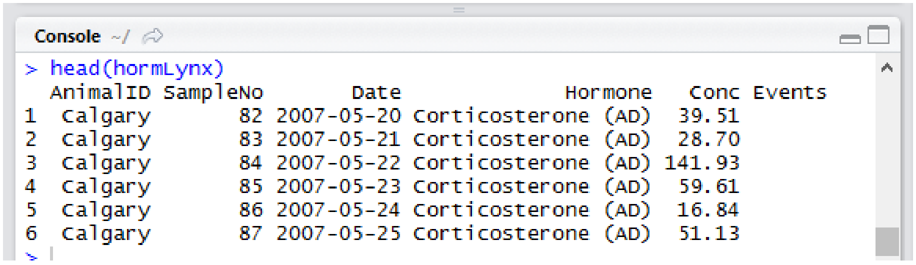
hormPlotOverlap()	Overlays multiple hormones on the same graph.	
hormPlotRatio()	Creates plots showing the ratio between 2 hormones.	
hormElephant	Example dataset including progesterone and cortisol data for elephants.	
hormLynx	Example dataset comparing the performance of 3 assays for monitoring ACTH challenge in Canada lynx.	

<u>Example Code</u>

General code

```
#-----  SET-UP  -----#
#Install associated packages - only run the first time#
install.packages('devtools', 'Rcurl')

#Load devtools and install hormLong - run first time or when hormLong updated#
library(devtools)
install_github('bfanson/hormLong')

#Load hormLong - run every time#
library(hormLong)


#-----  IMPORT & FORMAT DATA  -----#
#Import data#
ds <- hormRead()

#Format date#
ds <- hormDate(data = ds, date_var = 'Date', time_var = 'Time',
               name = 'Date1', date_order = 'mdy')

#Log transform conc#
ds$logconc <- log10(ds$Conc+1)
```

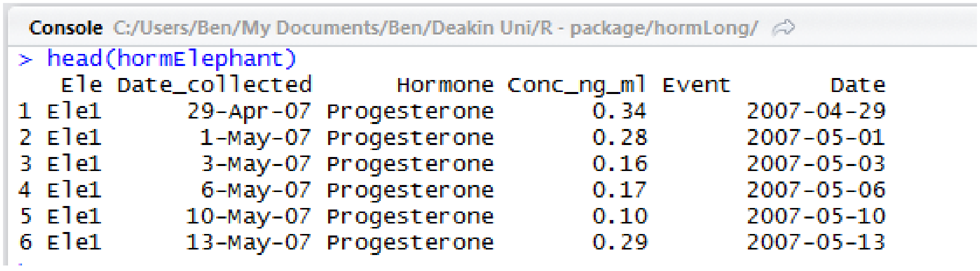
Example code using hormLynx dataset



```
> head(hormLynx)
  AnimalID SampleNo       Date          Hormone   Conc Events
1  Calgary       82 2007-05-20 Corticosterone (AD)  39.51
2  Calgary       83 2007-05-21 Corticosterone (AD)  28.70
3  Calgary       84 2007-05-22 Corticosterone (AD) 141.93
4  Calgary       85 2007-05-23 Corticosterone (AD)  59.61
5  Calgary       86 2007-05-24 Corticosterone (AD)  16.84
6  Calgary       87 2007-05-25 Corticosterone (AD)  51.13
```

```
#Run baseline calculation#
base <- hormBaseline(data = hormLynx, by_var = 'AnimalID, Hormone',
                     conc_var = 'Conc', time_var = 'datetime',
                     event_var = 'Events')

#Plot baseline data#
hormPlot(x = base, plot_per_page = 3, plot_height = 3)

#Get summary stats#
hormSumTable(x = base)

#Calculate and graph area under the curve (AUC)#
hormArea(x = base, lower_bound = 'peak', plot_per_page = 3,
         plot_height = 3)

#Compare individuals#
hormBoxplot(data = hormLynx, conc_var = 'Conc', id_var = 'AnimalID',
            by_var = 'Hormone', plot_height = 4, plot_width = 4,
            plot_per_page = 1)
```

Example code using hormElephant dataset


Console C:/Users/Ben/My Documents/Ben/Deakin Uni/R - package/hormLong/

```
> head(hormElephant)
   Ele Date_collected      Hormone Conc_ng_ml Event       Date
1 Ele1       29-Apr-07 Progesterone       0.34       2007-04-29
2 Ele1        1-May-07 Progesterone       0.28       2007-05-01
3 Ele1        3-May-07 Progesterone       0.16       2007-05-03
4 Ele1        6-May-07 Progesterone       0.17       2007-05-06
5 Ele1       10-May-07 Progesterone       0.10       2007-05-10
6 Ele1       13-May-07 Progesterone       0.29       2007-05-13
```

```
#Run baseline calculation#
ebase <- hormBaseline(data = hormElephant, by_var = 'Ele, Hormone',
                      conc_var = 'Conc_ng_ml', time_var = 'Date',
              event_var = 'Event')

#Plot data with break in x-axis#
hormPlotBreaks(x = ebase, break_cutoff = 35, break_buffer = 40,
               plot_per_page = 2, plot_height = 4)

#Looking at 2 hormones#
hormPlotOverlap(x = ebase, hormone_var = 'Hormone',
                date_format = '%b%y', colors = 'green,purple')

#Hormone ratios#
hormPlotRatio(x = ebase, hormone_var = 'Hormone',
              hormone_num = 'Cortisol',
              hormone_denom = 'Progesterone')
```

Using this manual

R code

R code is shown in light grey text boxes and can pasted directly into R:

```
#Basic code#
   ds <- hormDate(data = ds, date_var = 'Date')

#Full code#
   ds <- hormDate(data = ds, date_var = 'Date', time_var = 'Time',
   name = 'Date1', date_order = 'mdy')
```

Key

#Green	Green text preceded by '#' indicates comments that are not read by R
Blue	Blue text indicates information that is unique to your dataset. You can't just copy and paste this text – you need to be sure it matches your data.
Basic vs Full	'Basic code' indicates the bare minimum that you need to enter for the code to run. Default values will be used for all non-specified arguments. 'Full code' lists all the arguments available for a function and gives examples of information required for argument.

Notes

Helpful hints are shown in orange text boxes.

***Note:** Be sure you have loaded the package *hormLong* [run the line 'library(hormLong)']

Formatting your data

Purpose

Prior to importing your data into R, you need to be sure that

- Data are in univariate (or long) form
- Date (and optional time) variables are properly formatted
- Events are properly identified
- File is saved as a *csv* file
- We recommend simple headers (no spaces, no special characters (e.g. #/-?), not too long)

Instructions

1. Data are in long form

   Data needs to be organized such that all grouping variables are incorporated in the dataset and *not* individual column headers. For example, if you have multiple individuals in your dataset, then there should be a single column titled 'AnimalID' (as opposed to creating a separate column for each individual).

   This applies to multiple hormones as well. If there are multiple hormones in your dataset, then there should be 1 column titled 'Hormone' (where the specific hormone is listed) and 1 column titled 'Conc' (which gives the concentration for that sample and hormone). You should *not* have separate columns for each hormone (see Example below).

2. Format date

   Be sure that the "date" column includes ALL three date elements (day, month, year) and that you know the order.

   The default order is day-month-year (e.g. 01-Jan-2014). It does not matter which separator is used, whether month is abbreviated or full, or if year is 2- or 4- digits. If you wish to use a different order than the default, you will just need to specify the order of the date elements (see ***Importing data*** below).

3. Format time

   Time needs to be in military format (h:mm:ss; e.g., 13:30:55).

4. Events

   If there are events you want noted on the graphs, then be sure those events are added to the file. ***Do not*** create a new date column – all dates should be in the single 'Date' column formatted above. In the 'Event' column, list the event as you would like it to appear on the graph (we suggest short names).

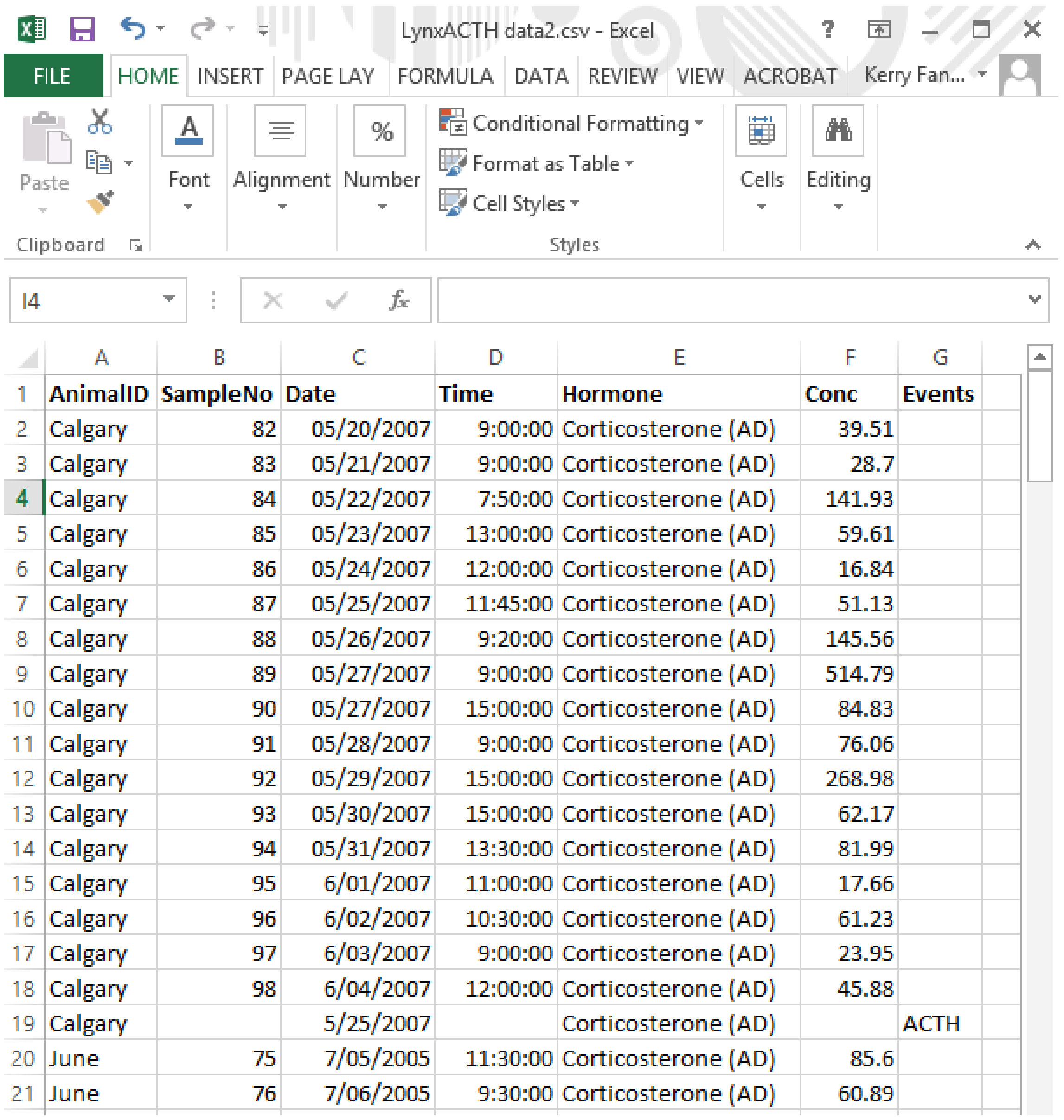
Events do not need to be in order and dates can be duplicated. Therefore, we suggest that the easiest way to incorporate event information is to add it to the bottom of the data table. Be sure that all necessary grouping information is included for each event.

If you have multiple hormones in your dataset and you want the event to appear on the graph for each hormone, then create multiple entries for the event, with each entry specifying a different hormone.

5. Save as *csv* file

The import is designed for *csv* files. If you select "Save As", there is an option to specify file type (below 'File name'). Select '**CSV (Comma delimited) (*.csv)**'. It will warn you that you may lose some formatting, which is OK.

Example



	A	B	C	D	E	F	G
1	AnimalID	SampleNo	Date	Time	Hormone	Conc	Events
2	Calgary	82	05/20/2007	9:00:00	Corticosterone (AD)	39.51	
3	Calgary	83	05/21/2007	9:00:00	Corticosterone (AD)	28.7	
4	Calgary	84	05/22/2007	7:50:00	Corticosterone (AD)	141.93	
5	Calgary	85	05/23/2007	13:00:00	Corticosterone (AD)	59.61	
6	Calgary	86	05/24/2007	12:00:00	Corticosterone (AD)	16.84	
7	Calgary	87	05/25/2007	11:45:00	Corticosterone (AD)	51.13	
8	Calgary	88	05/26/2007	9:20:00	Corticosterone (AD)	145.56	
9	Calgary	89	05/27/2007	9:00:00	Corticosterone (AD)	514.79	
10	Calgary	90	05/27/2007	15:00:00	Corticosterone (AD)	84.83	
11	Calgary	91	05/28/2007	9:00:00	Corticosterone (AD)	76.06	
12	Calgary	92	05/29/2007	15:00:00	Corticosterone (AD)	268.98	
13	Calgary	93	05/30/2007	15:00:00	Corticosterone (AD)	62.17	
14	Calgary	94	05/31/2007	13:30:00	Corticosterone (AD)	81.99	
15	Calgary	95	6/01/2007	11:00:00	Corticosterone (AD)	17.66	
16	Calgary	96	6/02/2007	10:30:00	Corticosterone (AD)	61.23	
17	Calgary	97	6/03/2007	9:00:00	Corticosterone (AD)	23.95	
18	Calgary	98	6/04/2007	12:00:00	Corticosterone (AD)	45.88	
19	Calgary		5/25/2007		Corticosterone (AD)		ACTH
20	June	75	7/05/2005	11:30:00	Corticosterone (AD)	85.6	
21	June	76	7/06/2005	9:30:00	Corticosterone (AD)	60.89	

Getting started with R

Overview

- Install R
- Install R editor
- A few R basics

Instructions

1. Install R

   R is a free software program that is widely used for statistical analysis and graphing. It allows users to create packages that are then available to all users. R can be downloaded from the following website:

   http://cran.r-project.org/

2. Install R editor

   There are a wide number of R editors available that make it easier to interface with R. With these programs, everything is done through the editor, which then interacts with R "behind-the-scenes" once commands are sent. One of the most popular R editors is 'RStudio' which can be downloaded here:

   http://www.rstudio.com/products/rstudio/download/

Note: For a great overview of R and RStudio – including downloading them and getting started – we recommend the following video. It's pretty short and after watching it you should have the necessary basics for using the package hormLong.
https://www.youtube.com/watch?v=lVKMsaWju8w

3. R basics – Terminology

   Package – an R package is a program that has been written to carry out specific functions. Packages need to be installed on your computer and loaded into R before you can work with them.

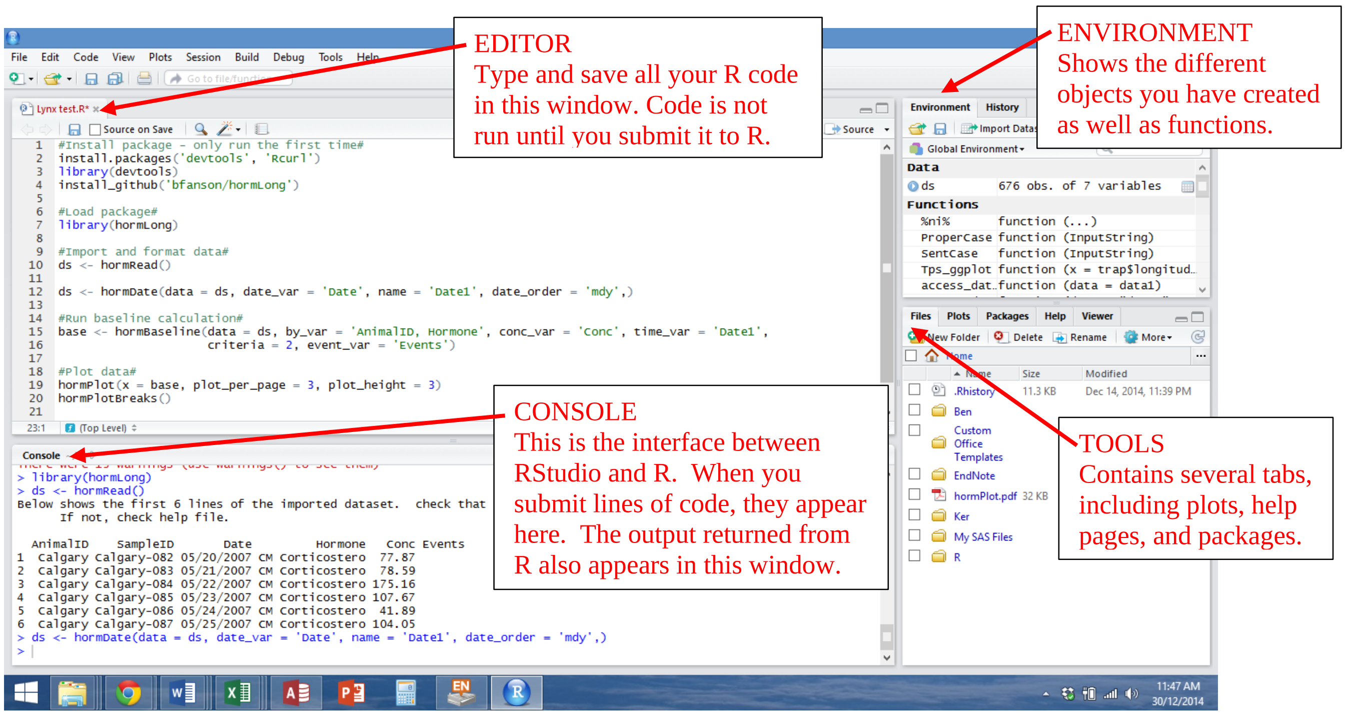
   Function – R packages generally consist of several functions that do different tasks (e.g. average() and stdev() in Excel are functions). To execute a function, you need to call the function name and specify any arguments needed to run that task.

   Argument – Functions can be tailored to meet your needs by specifying certain details. These arguments always go in parentheses following the function name.

   Object – You can create different objects in R, the most common of which is data tables. *If you assign a name to an object*, then it is stored in the program memory and you can call that object for subsequent functions. To assign a name, type the name followed by "<-" and then the details of the object you want to create or import.

4. R basics – Finding your way around

When you open RStudio, there are 4 different windows/sections that appear: editor, console, environment, and tools.



EDITOR
Type and save all your R code in this window. Code is not run until you submit it to R.

```
#Install package - only run the first time#
install.packages('devtools', 'Rcurl')
library(devtools)
install_github('bfanson/hormLong')

#Load package#
library(hormLong)

#Import and format data#
ds <- hormRead()

ds <- hormDate(data = ds, date_var = 'Date', name = 'Date1', date_order = 'mdy',)

#Run baseline calculation#
base <- hormBaseline(data = ds, by_var = 'AnimalID, Hormone', conc_var = 'Conc', time_var = 'Date1',
                     criteria = 2, event_var = 'Events')

#Plot data#
hormPlot(x = base, plot_per_page = 3, plot_height = 3)
hormPlotBreaks()
```

ENVIRONMENT
Shows the different objects you have created as well as functions.

```
Data
ds            676 obs. of 7 variables
Functions
  %ni%        function (...)
  ProperCase  function (InputString)
  SentCase    function (InputString)
  Tps_ggplot  function (x = trap$longitud…
  access_dat… function (data = data1)
```

CONSOLE
This is the interface between RStudio and R. When you submit lines of code, they appear here. The output returned from R also appears in this window.

```
> library(hormLong)
> ds <- hormRead()
Below shows the first 6 lines of the imported dataset.  check that
      If not, check help file.

  AnimalID    SampleID       Date       Hormone    Conc Events
1  Calgary Calgary-082 05/20/2007 CM Corticostero  77.87
2  Calgary Calgary-083 05/21/2007 CM Corticostero  78.59
3  Calgary Calgary-084 05/22/2007 CM Corticostero 175.16
4  Calgary Calgary-085 05/23/2007 CM Corticostero 107.67
5  Calgary Calgary-086 05/24/2007 CM Corticostero  41.89
6  Calgary Calgary-087 05/25/2007 CM Corticostero 104.05
> ds <- hormDate(data = ds, date_var = 'Date', name = 'Date1', date_order = 'mdy',)
> 
```

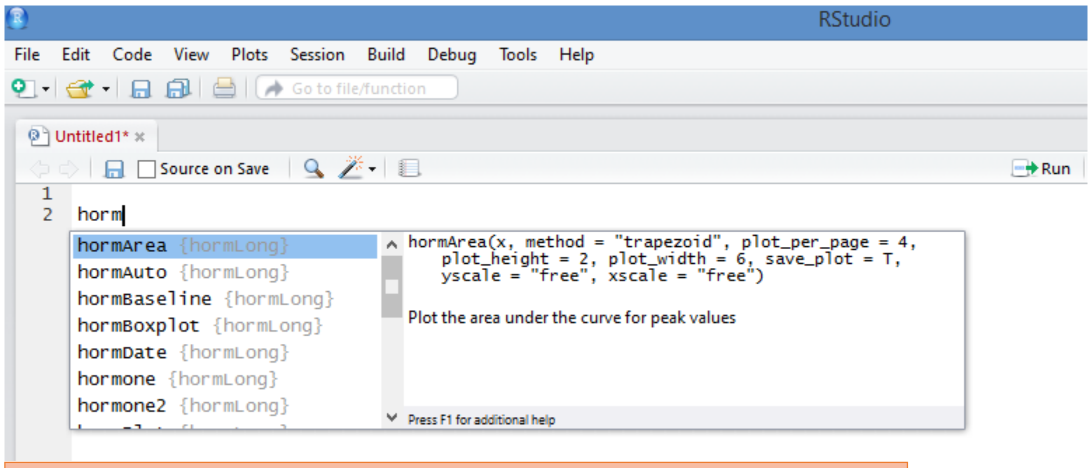
TOOLS
Contains several tabs, including plots, help pages, and packages.

```
Name              Size     Modified
.Rhistory         11.3 KB  Dec 14, 2014, 11:39 PM
Ben
Custom Office Templates
EndNote
hormPlot.pdf      32 KB
Ker
My SAS Files
R
```

5. R basics – Autofill

TAB is useful for getting more information about functions.

If you don't know the exact **name of the function** but you know what it starts with
→ type the first few letters and hit 'TAB'
It will present a list of all functions that start with those letters, as well as a description of what the function does. Use arrow keys to scroll through functions.


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Untitled1*
Source on Save
Run
Press F1 for additional help

```
horm
```

```
hormArea {hormLong}
hormAuto {hormLong}
hormBaseline {hormLong}
hormBoxplot {hormLong}
hormDate {hormLong}
hormone {hormLong}
hormone2 {hormLong}
```

```
hormArea(x, method = "trapezoid", plot_per_page = 4,
    plot_height = 2, plot_width = 6, save_plot = T,
    yscale = "free", xscale = "free")
```

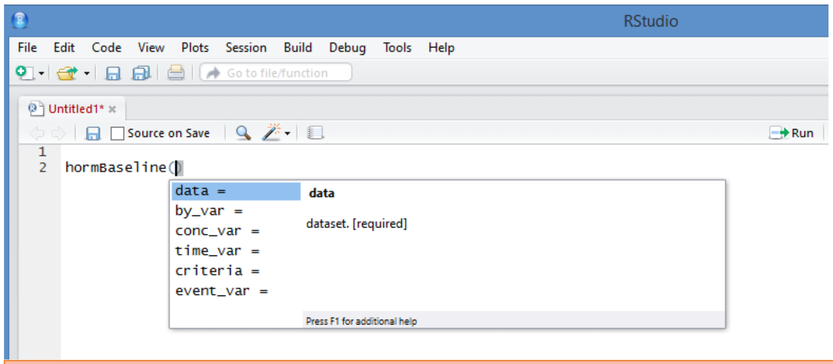
Plot the area under the curve for peak values

*Note: All of the functions associated with the hormLong package start with 'horm'

If you don't remember the **arguments** for a particular function

→ type the function name and open the parentheses, then hit 'TAB'

It will present a list of all the arguments for that function, as well as a description of what information is needed and whether the argument is required or optional. Arrow keys can be used to scroll through arguments.



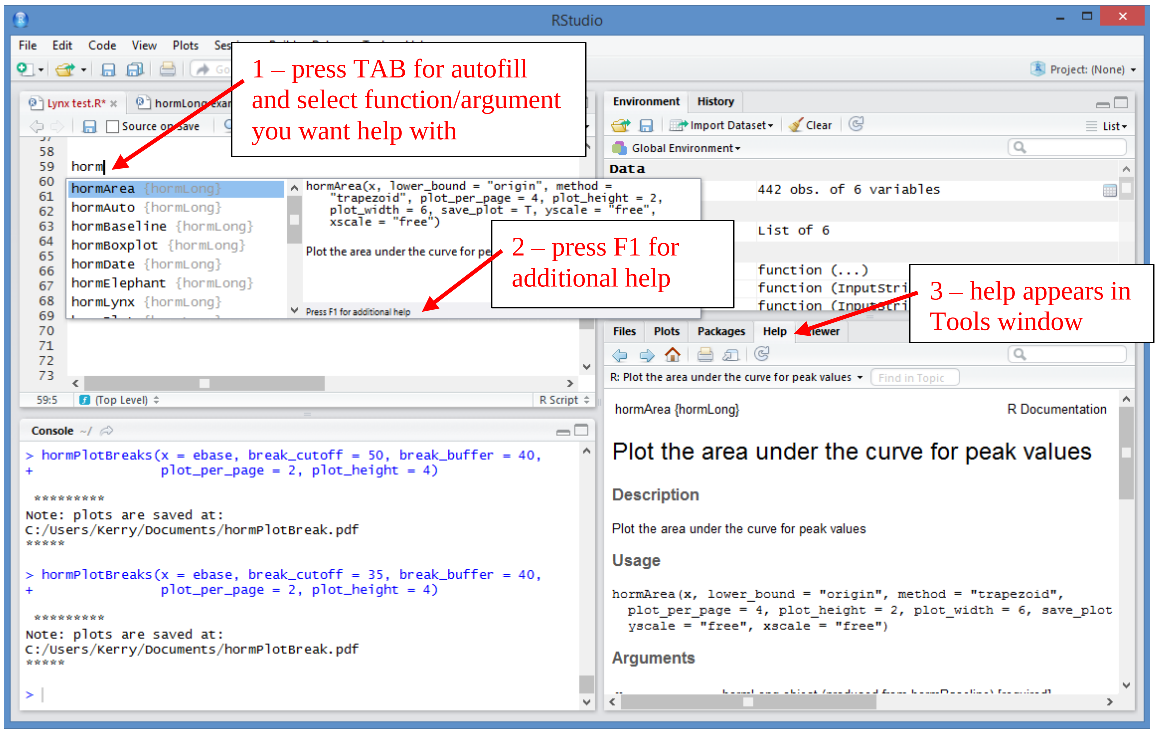
```
hormBaseline()
```

```
data =          data
by_var =
conc_var =      dataset. [required]
time_var =
criteria =
event_var =
                Press F1 for additional help
```

Note: For all arguments in *hormLong* functions, we specify in the description whether it is required, optional, or required but there is a default set.

6. R basics – Getting help

TAB then **F1** can be used to get more help on any of the functions or arguments.


RStudio
File Edit Code View Plots
Lynx test.R*
hormLong exam
Source on Save
1 – press TAB for autofill and select function/argument you want help with
2 – press F1 for additional help
3 – help appears in Tools window
Environment History
Import Dataset
Clear
Global Environment
List
Project: (None)
Files Plots Packages Help Viewer
Find in Topic
(Top Level)
R Script
Console ~/

```
horm
```

```
hormArea      {hormLong}    hormArea(x, lower_bound = "origin", method =
hormAuto      {hormLong}        "trapezoid", plot_per_page = 4, plot_height = 2,
hormBaseline  {hormLong}        plot_width = 6, save_plot = T, yscale = "free",
hormBoxplot   {hormLong}        xscale = "free")
hormDate      {hormLong}
hormElephant  {hormLong}    Plot the area under the curve for pe
hormLynx      {hormLong}
                            Press F1 for additional help
```

```
Data
                442 obs. of 6 variables

                List of 6

                function (...)
                function (InputStri
                function (InputStri
```

```
> hormPlotBreaks(x = ebase, break_cutoff = 50, break_buffer = 40,
+                plot_per_page = 2, plot_height = 4)

 *********
Note: plots are saved at:
C:/Users/Kerry/Documents/hormPlotBreak.pdf
*****

> hormPlotBreaks(x = ebase, break_cutoff = 35, break_buffer = 40,
+                plot_per_page = 2, plot_height = 4)

 *********
Note: plots are saved at:
C:/Users/Kerry/Documents/hormPlotBreak.pdf
*****

> |
```

R: Plot the area under the curve for peak values

hormArea {hormLong} R Documentation

Plot the area under the curve for peak values

Description

Plot the area under the curve for peak values

Usage

```
hormArea(x, lower_bound = "origin", method = "trapezoid",
  plot_per_page = 4, plot_height = 2, plot_width = 6, save_plot
  yscale = "free", xscale = "free")
```

Arguments

You can also submit the following code: '?function_name' (e.g. ?hormPlot).

7. R basics – Submitting commands

   The main thing you need to know in R is how to submit commands. There are several ways to do this:

   - Highlight the code to submit and hit the Run button (top right corner of Editor screen)
   - Place cursor anywhere on the line you want to submit (if only 1 line) and hit Ctrl+Enter
   - Highlight the code and hit Ctrl+Enter

8. R basics – Annotating your code

   The **#** sign can be used to comment-out code. This is useful if you want to make notes in your code about what particular steps are for, but don't necessarily want it submitted to R.

Getting started with hormLong

Purpose

- Install hormLong and associated packages
- Load 'hormLong'

Code

```
#Install associated packages - only run the first time
 #needs internet connection#
   install.packages('devtools', 'Rcurl')

#Load devtools and install hormLong - run first time or when hormLong updated
 #needs internet connection#
   library(devtools)
   install_github('bfanson/hormLong')

#Load package#
   library(hormLong)
```

Reminder: *green text preceded by '#' indicates comments – text will not be submitted to R*

Instructions

1. Installing associated packages

   There are a few packages that hormLong depends on, so those need to be installed first (devtools, Rcurl). This only needs to be done once on your computer.

2. Installing hormLong

   The package hormLong can be installed from github. In order for the download to work, you need to load the devtools package. [*Cut and paste the code above into RStudio.*]
   This code only needs to be run the first time you use the package, or after any upgrades to the package.

3. Working with hormLong

   Each time you want to use the package, you must load the package [*library(hormLong)*].

hormRead() & hormDate() – Importing data

**Note:* Be sure you have loaded the package *hormLong*
[run the line 'library(hormLong)']

Purpose

hormRead() helps you import your data into R. This function will open a dialog box and you just need to select the file you want to import. You also need to assign the dataset an object name – this will save the dataset in the memory so that you can call on it for other functions.

hormDate() formats the date. Since dates can be formatted in so many different ways, there can sometimes be problems working with dates. This function standardizes everything into a uniform format.
In addition, if you have a time variable (e.g., multiple samples collected in a day), this function combines date and time into a single date-time variable. Since this package is designed for longitudinal hormone monitoring, this date-time variable allows everything to be assigned in chronological order.

Overview

INPUT:	Excel spreadsheet saved in *csv* format
ACTIONS:	Import data into R [*hormRead()*] Fix date format [*hormDate()*]
OUTPUT:	An R data object that is formatted and ready to use

Code

```
#Import data#
   ds <- hormRead()

#Format date – Basic code#
   ds <- hormDate(data = ds, date_var = 'Date')

#Format date – Full code#
   ds <- hormDate(data = ds, date_var = 'Date', time_var = 'Time',
   name = 'Date1', date_order = 'mdy')

#Log-transform data (optional)#
   ds$logconc <- log10(ds$Conc+1)
```

**Reminder*: *Blue text indicates information unique to your dataset – should be modified accordingly*

Instructions

Import data

1. Identify the object name, or how you want to refer to the dataset (e.g., ‘ds’, ‘dataset’, ‘ACTH’)
2. Type ‘ <- ’ [This lets R know that the information before this is the object name and the information after this is the information assigned to this object.]

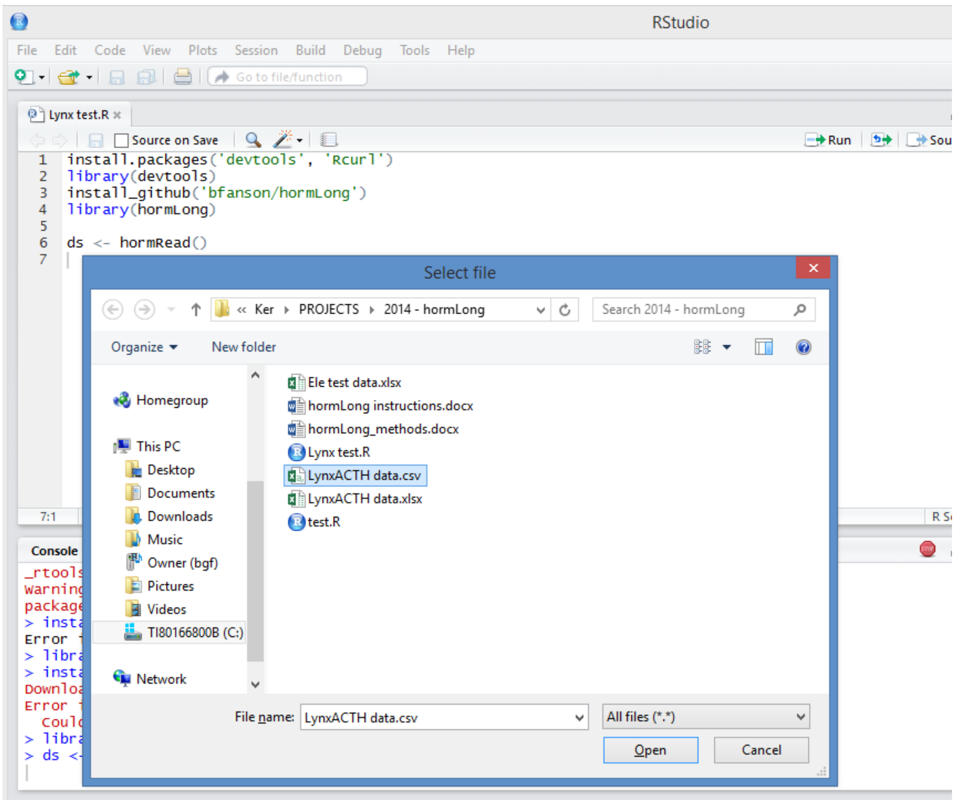
3. Use the function *hormRead()* to identify and read-in your dataset

EX.

ds <- hormRead()

Object name | joiner | Information assigned to object

4. Once you submit this line of code (with nothing in the parentheses), it will open a dialog box asking you to select the *csv* file that you want to import. After you select the file, it will be imported into R and assigned as an object.



```
install.packages('devtools', 'Rcurl')
library(devtools)
install_github('bfanson/hormLong')
library(hormLong)

ds <- hormRead()
```

Ele test data.xlsx
hormLong instructions.docx
hormLong_methods.docx
Lynx test.R
LynxACTH data.csv
LynxACTH data.xlsx
test.R

File name: LynxACTH data.csv
All files (*.*)

> ***Note:*** Sometimes the ‘Select file’ dialog box will appear behind the RStudio window and it is not apparent that it is there. If you do not see the dialog box, minimize the RStudio window and it should be there.

Fix date format

1. Specify the name of the new dataset (you can also keep the same name as the original dataset)
2. Assign the *hormDate()* function
3. Specify the function arguments

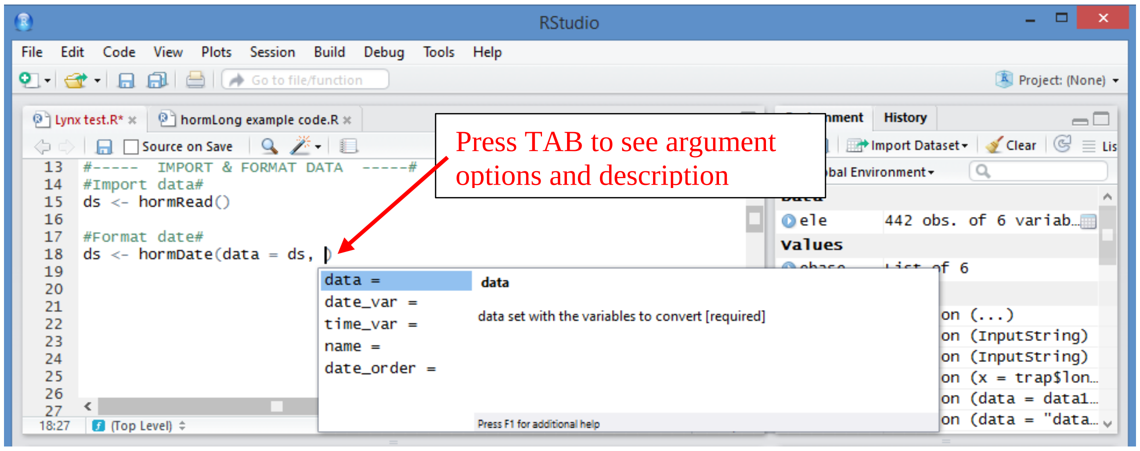
   a. **data** – name of the dataset
   Required

   b. **date_var** – name of the date column
   Required
   The name of the column needs to be in quotes. As mentioned in section 2, your data should have a standard date format (the specific format does not matter as long as it is recognizable as a date). Your data should be in long form, so you should not have more than 1 date column.

   c. **time_var** – name of the time column
   Required only if you have a time variable
   The name of the column needs to be in quotes. As mentioned in section 2, this column needs to be in 24-hour (military) time format.

   d. **name** – name of the new date (or datetime) column
   Default is set to 'datetime'
   If you want a different variable name, specify that here. The name of the column needs to be in quotes. This column will be the single time variable used in all subsequent analyses and will be the x-axis on most graphs. If you have date and time, then it will merge those two fields into one; otherwise it will just include date in a standardized format.

   e. **date_order** – specifies order of date components in *csv* file
   Default is set as 'dmy' (day, month, year)
   If your date is not formatted in 'dmy' order, then you need to specify the order in which the date components appear in your *csv* file (e.g., 'mdy', 'ymd'). The order needs to be in quotes.



```
#-----  IMPORT & FORMAT DATA  -----#
#Import data#
ds <- hormRead()

#Format date#
ds <- hormDate(data = ds, )
```

```
data =          data
date_var =
time_var =      data set with the variables to convert [required]
name =
date_order =
                Press F1 for additional help
```

```
ele     442 obs. of 6 variab...
Values
```

Note: If your time variable is in count form (e.g., days from ACTH challenge) rather than date form, you do not need to run *hormDate()*. For all other functions (except *hormPlotBreaks()*), you can call your day variable in the same way as the datetime variable created here.

Log-transform concentration data

1. Use the code

   ds$logconc <- log10(ds$Conc+1)

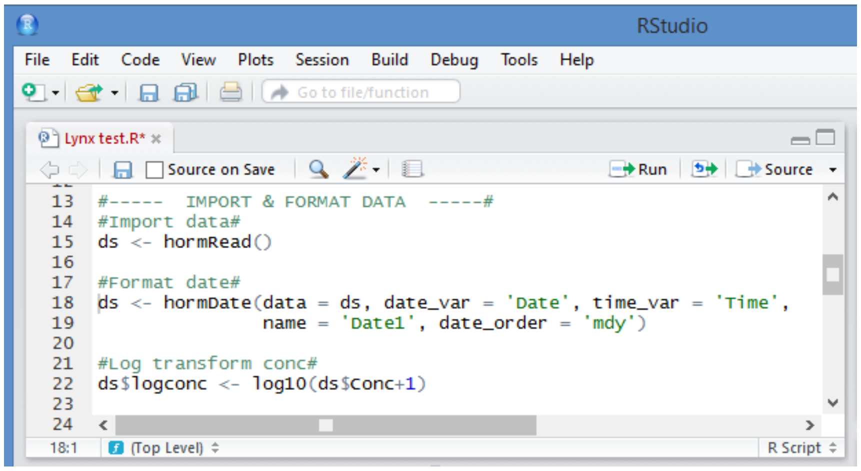
   Be sure you modify the name of your dataset before the $ sign (in this case 'ds') corresponds to the name you assigned your dataset in the previous step. Also be sure you adjust the name of your raw concentration variable (in this case 'Conc') accordingly.

2. Option 2 is to log-transform your data in Excel and save as a *csv* file.

Example

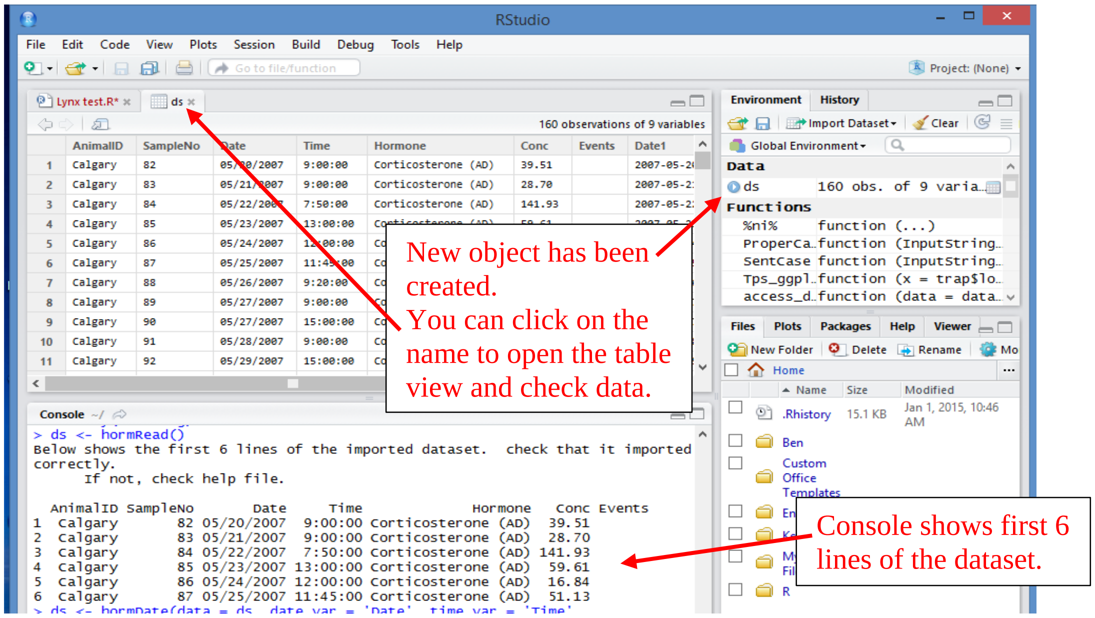
DATA: hormLynx

CODE:



```
#-----  IMPORT & FORMAT DATA  -----#
#Import data#
ds <- hormRead()

#Format date#
ds <- hormDate(data = ds, date_var = 'Date', time_var = 'Time',
                name = 'Date1', date_order = 'mdy')

#Log transform conc#
ds$logconc <- log10(ds$Conc+1)
```

OUTPUT:



160 observations of 9 variables

	AnimalID	SampleNo	Date	Time	Hormone	Conc	Events	Date1
1	Calgary	82	05/20/2007	9:00:00	Corticosterone (AD)	39.51		2007-05-2
2	Calgary	83	05/21/2007	9:00:00	Corticosterone (AD)	28.70		2007-05-2
3	Calgary	84	05/22/2007	7:50:00	Corticosterone (AD)	141.93		2007-05-2
4	Calgary	85	05/23/2007	13:00:00	Co			
5	Calgary	86	05/24/2007	12:00:00	Co			
6	Calgary	87	05/25/2007	11:45:00	Co			
7	Calgary	88	05/26/2007	9:20:00	Co			
8	Calgary	89	05/27/2007	9:00:00	Co			
9	Calgary	90	05/27/2007	15:00:00	Co			
10	Calgary	91	05/28/2007	9:00:00	Co			
11	Calgary	92	05/29/2007	15:00:00	Co			

```
Data
ds          160 obs. of 9 varia…
Functions
%ni%        function (...)
ProperCa…   function (InputString…
SentCase    function (InputString…
Tps_ggpl…   function (x = trap$lo…
access_d…   function (data = data…
```

	Name	Size	Modified
	.Rhistory	15.1 KB	Jan 1, 2015, 10:46 AM

New object has been created.
You can click on the name to open the table view and check data.

```
> ds <- hormRead()
Below shows the first 6 lines of the imported dataset.  check that it imported
correctly.
      If not, check help file.

  AnimalID SampleNo       Date     Time              Hormone   Conc Events
1  Calgary       82 05/20/2007  9:00:00 Corticosterone (AD)  39.51
2  Calgary       83 05/21/2007  9:00:00 Corticosterone (AD)  28.70
3  Calgary       84 05/22/2007  7:50:00 Corticosterone (AD) 141.93
4  Calgary       85 05/23/2007 13:00:00 Corticosterone (AD)  59.61
5  Calgary       86 05/24/2007 12:00:00 Corticosterone (AD)  16.84
6  Calgary       87 05/25/2007 11:45:00 Corticosterone (AD)  51.13
> ds <- hormDate(data = ds, date_var = 'Date', time_var = 'Time'
```

Console shows first 6 lines of the dataset.

Example datasets

Purpose

hormLynx and ***hormElephant*** are example datasets that are used throughout this manual and in the accompanying examples. These example datasets help ensure that you test your code and make sure your program is working by checking them against our results.

Overview

hormLynx includes fecal glucocorticoid metabolite data for 3 Canada lynx. Lynx underwent an ACTH challenge for biological validation of the assay. For each individual, the performance of 3 different assays was compared. See (Fanson et al. 2012).

Column Name	**Description [*Values*]**
AnimalID	Individual lynx identifier [*Calgary, June, Lexi*]
SampleNo	Sample number for each individual
Date	Date sample collected
Time	Time sample collected
Hormone	Name of assay used [*Corticosterone (AD), Corticosterone (CJM), Cortisol (CJM)*]
Conc	Final concentration of FGM (ng/g)
Events	Important events [*ACTH*]
datetime	Merged date and time in standardized format

hormElephant includes serum cortisol and progesterone data for 2 Asian elephants. Patterns were monitored with respect to stage of estrous cycle and pregnancy. See (Fanson et al. 2014).

Column Name	**Description [*Values*]**
Ele	Individual elephant identifier [*Ele1, Ele2*]
Date_collected	Date sample collected
Hormone	Hormone type [*Progesterone, Cortisol*]
Conc_ng_ml	Final hormone concentration (ng/ml)
Events	Important events [*Mating, Birth*]
Date	Date in standardized format

Code

```
#Use example dataset in a function#
    base <- hormBaseline(data = hormLynx)

#View first 6 lines of dataset#
    head(hormLynx)

#Open dataset in new viewing window#
    View(hormLynx)
```

Instructions

1. To access these datasets, you call them the same way you would any dataset.
   E.g., hormBaseline(data = hormLynx)

2. To see examples of code using these datasets, open the help files (TAB then F1). You can copy and paste that code directly into the editor window and it should run.

3. You can also follow examples using these datasets and view the corresponding output throughout this manual (see Examples at the end of each section).

hormBaseline() – Calculating baseline

Purpose

hormBaseline() performs an iterative baseline calculation on hormone data based on the mean and standard deviation (SD) of the data. This approach is commonly used to identify peaks in longitudinal hormone datasets. Baseline is calculated by excluding points greater than the mean+(*n**SD) and repeating this process until no more points exceed this threshold (Clifton and Steiner 1983, Brown et al. 1996).

The value of *n* (i.e., the number of standard deviations) can be adjusted depending on the nature of the dataset. The criterion most often used is 2*SD, but the appropriateness of the criteria depends on (1) number of samples and (2) amount of variability in the baseline. Increasing the criteria (e.g., 3*SD) will target bigger peaks and yields a more conservative approach. However, if you have a small dataset or lots of variability in the baseline, you may miss some peaks. If you decrease the criteria (e.g., 1.5*SD), then you are more likely to include small peaks or even blips in the baseline. Graph your data with different criteria to see which is the most appropriate for your data.

Overview

INPUT:	R data object created from imported data
ACTIONS:	Run iterative baseline calculation on hormone data
OUTPUT:	An R object (specifically a hormLong object) A *csv* file (hormBaseData.csv) with column identifying peaks

Code

```
#Calculate baseline – Basic code#
   base <- hormBaseline(data = ds, conc_var = 'Conc', time_var = 'Date1')

#Calculate baseline – Full code#
   base <- hormBaseline(data = ds, by_var = 'AnimalID, Hormone',
   conc_var = 'Conc', time_var = 'Date1', criteria = 1.5, event_var = 'Events')
```

Instructions

1. Name a new object
2. Assign the *hormBaseline()* function
3. Specify function arguments

    a. **data** – name of the dataset
    Required

    b. **by_var** – specifies any grouping variables
    Optional

If you have multiple species, individuals, or hormones in your dataset but you want to calculate baseline separately for each group, you can group by those factors. Column names of grouping variables must be in quotes with a comma separating them (e.g., 'AnimalID, Hormone').

c. **conc_var** – name of the hormone concentration column
Required

d. **time_var** – name of the date, datetime, or day column
Required

e. **criteria** – number of SD used for the baseline calculation
Default set at 2
The iterative baseline approach excludes points that exceed the *mean+(SD*criteria)*. Here, you can determine the value of the criteria.

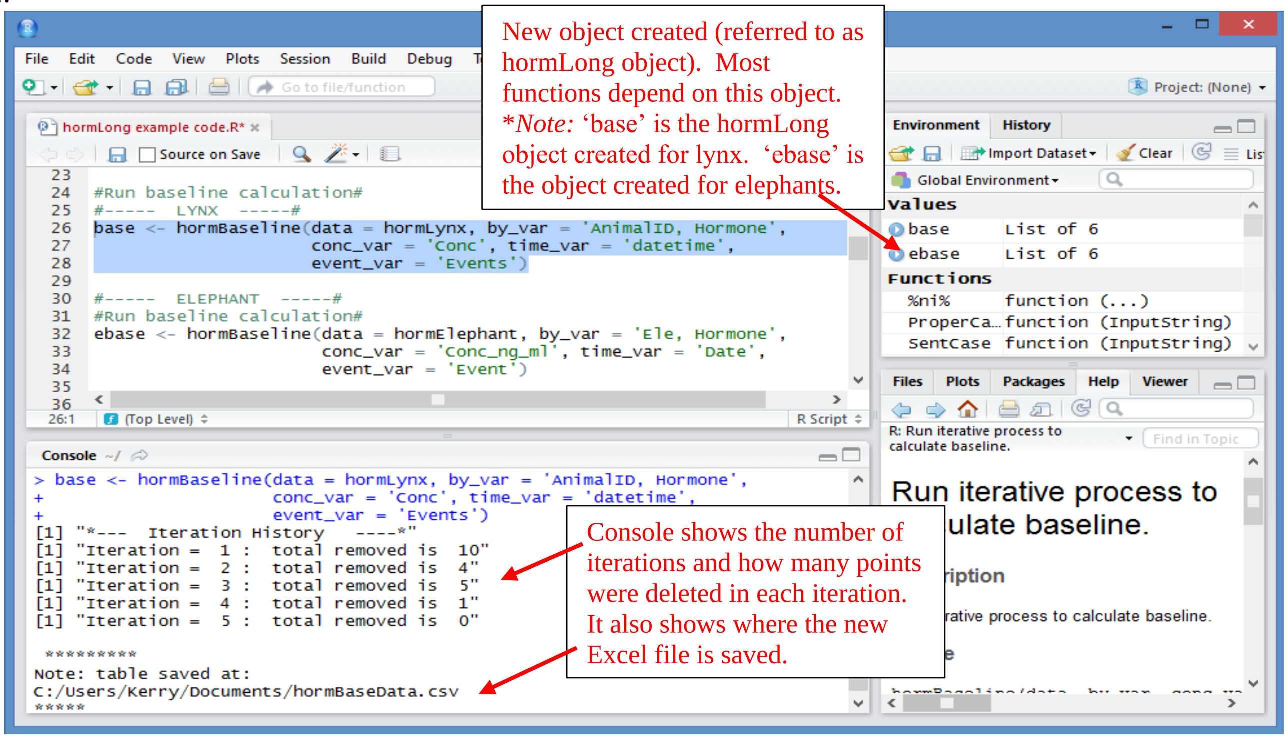
f. **event_var** – name of the event column
Optional
If you have events that you want plotted on the graphs, then specify the name of the event column here. The program will print any text that appears in this column on the appropriate day.

g. **save_data** – whether to save dataset as a *csv* file
Default set to TRUE

Example

DATA: hormLynx & hormElephant

CODE:



```
#Run baseline calculation#
#-----  LYNX  -----#
base <- hormBaseline(data = hormLynx, by_var = 'AnimalID, Hormone',
                     conc_var = 'Conc', time_var = 'datetime',
                     event_var = 'Events')

#-----  ELEPHANT  -----#
#Run baseline calculation#
ebase <- hormBaseline(data = hormElephant, by_var = 'Ele, Hormone',
                      conc_var = 'Conc_ng_ml', time_var = 'Date',
                      event_var = 'Event')
```

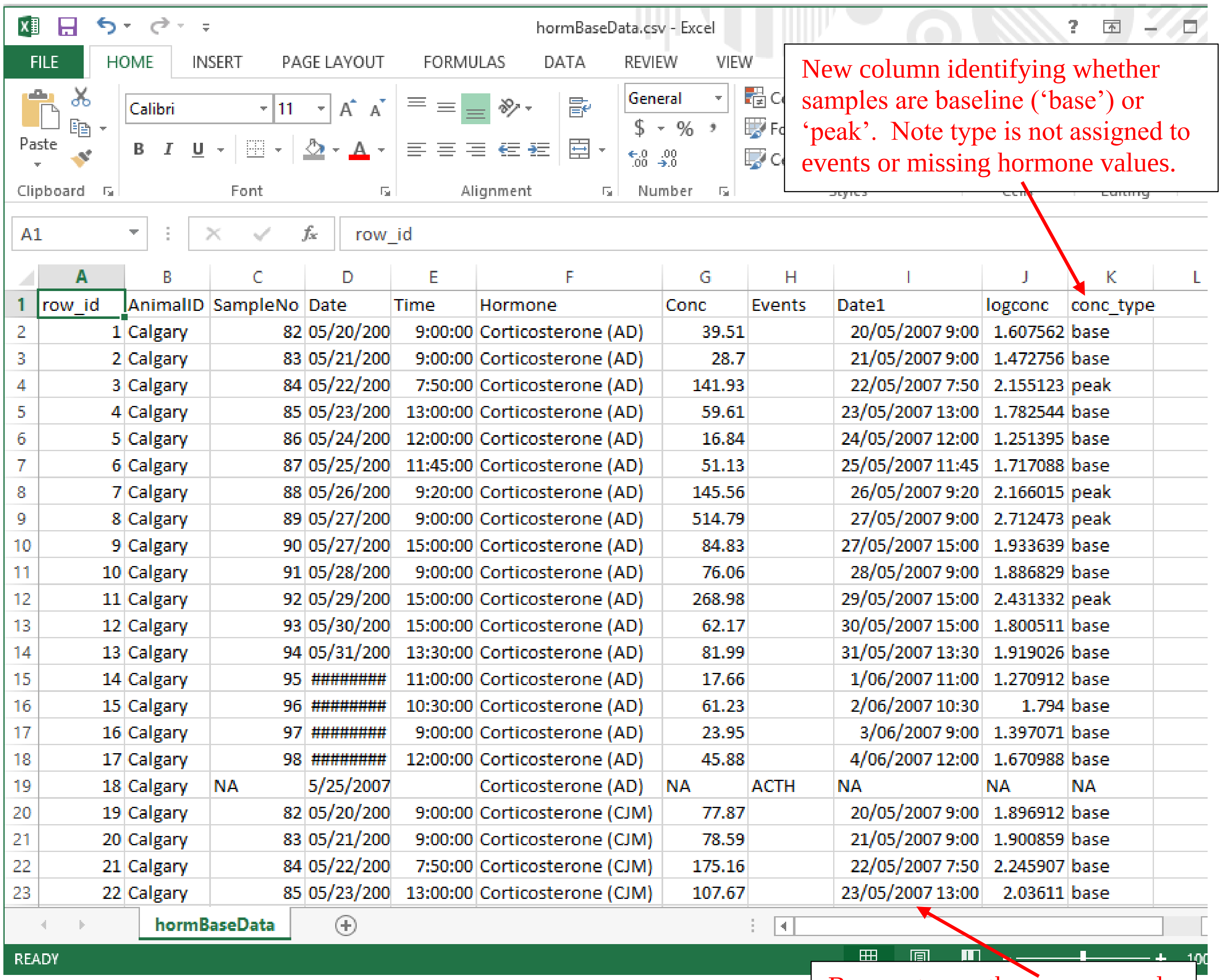
New object created (referred to as hormLong object). Most functions depend on this object. **Note:* 'base' is the hormLong object created for lynx. 'ebase' is the object created for elephants.

```
Values
base        List of 6
ebase       List of 6
Functions
%ni%        function (...)
ProperCa…   function (InputString)
SentCase    function (InputString)
```

```
Console ~/
> base <- hormBaseline(data = hormLynx, by_var = 'AnimalID, Hormone',
+                      conc_var = 'Conc', time_var = 'datetime',
+                      event_var = 'Events')
[1] "*---  Iteration History   ---*"
[1] "Iteration =  1 :  total removed is  10"
[1] "Iteration =  2 :  total removed is  4"
[1] "Iteration =  3 :  total removed is  5"
[1] "Iteration =  4 :  total removed is  1"
[1] "Iteration =  5 :  total removed is  0"

 *********
Note: table saved at:
C:/Users/Kerry/Documents/hormBaseData.csv
*****
```

Console shows the number of iterations and how many points were deleted in each iteration. It also shows where the new Excel file is saved.

OUTPUT:



	A	B	C	D	E	F	G	H	I	J	K
1	row_id	AnimalID	SampleNo	Date	Time	Hormone	Conc	Events	Date1	logconc	conc_type
2	1	Calgary	82	05/20/200	9:00:00	Corticosterone (AD)	39.51		20/05/2007 9:00	1.607562	base
3	2	Calgary	83	05/21/200	9:00:00	Corticosterone (AD)	28.7		21/05/2007 9:00	1.472756	base
4	3	Calgary	84	05/22/200	7:50:00	Corticosterone (AD)	141.93		22/05/2007 7:50	2.155123	peak
5	4	Calgary	85	05/23/200	13:00:00	Corticosterone (AD)	59.61		23/05/2007 13:00	1.782544	base
6	5	Calgary	86	05/24/200	12:00:00	Corticosterone (AD)	16.84		24/05/2007 12:00	1.251395	base
7	6	Calgary	87	05/25/200	11:45:00	Corticosterone (AD)	51.13		25/05/2007 11:45	1.717088	base
8	7	Calgary	88	05/26/200	9:20:00	Corticosterone (AD)	145.56		26/05/2007 9:20	2.166015	peak
9	8	Calgary	89	05/27/200	9:00:00	Corticosterone (AD)	514.79		27/05/2007 9:00	2.712473	peak
10	9	Calgary	90	05/27/200	15:00:00	Corticosterone (AD)	84.83		27/05/2007 15:00	1.933639	base
11	10	Calgary	91	05/28/200	9:00:00	Corticosterone (AD)	76.06		28/05/2007 9:00	1.886829	base
12	11	Calgary	92	05/29/200	15:00:00	Corticosterone (AD)	268.98		29/05/2007 15:00	2.431332	peak
13	12	Calgary	93	05/30/200	15:00:00	Corticosterone (AD)	62.17		30/05/2007 15:00	1.800511	base
14	13	Calgary	94	05/31/200	13:30:00	Corticosterone (AD)	81.99		31/05/2007 13:30	1.919026	base
15	14	Calgary	95	#########	11:00:00	Corticosterone (AD)	17.66		1/06/2007 11:00	1.270912	base
16	15	Calgary	96	#########	10:30:00	Corticosterone (AD)	61.23		2/06/2007 10:30	1.794	base
17	16	Calgary	97	#########	9:00:00	Corticosterone (AD)	23.95		3/06/2007 9:00	1.397071	base
18	17	Calgary	98	#########	12:00:00	Corticosterone (AD)	45.88		4/06/2007 12:00	1.670988	base
19	18	Calgary	NA	5/25/2007		Corticosterone (AD)	NA	ACTH	NA	NA	NA
20	19	Calgary	82	05/20/200	9:00:00	Corticosterone (CJM)	77.87		20/05/2007 9:00	1.896912	base
21	20	Calgary	83	05/21/200	9:00:00	Corticosterone (CJM)	78.59		21/05/2007 9:00	1.900859	base
22	21	Calgary	84	05/22/200	7:50:00	Corticosterone (CJM)	175.16		22/05/2007 7:50	2.245907	base
23	22	Calgary	85	05/23/200	13:00:00	Corticosterone (CJM)	107.67		23/05/2007 13:00	2.03611	base

New column identifying whether samples are baseline ('base') or 'peak'. Note type is not assigned to events or missing hormone values.

Be sure to use the new, properly formatted 'Date1' column, not the original 'Date' column.

hormPlot() – Making longitudinal plots

Purpose

hormPlot() is used to make longitudinal plots of your data, which is useful for visualizing trends over time. This function creates separate longitudinal graphs for each group identified by the *by_var* statement in *hormBaseline()* (e.g., different individuals, hormones, etc). In addition to the hormone data, plots can include (1) a horizontal reference line indicating the cutoff value for peaks, determined from baseline calculation above, and (2) arrows and text to indicate certain events. A number of formatting options are available to tailor the appearance of the graph.

Overview

INPUT:	Data object created by *hormBaseline()*
ACTIONS:	Create longitudinal graphs
OUTPUT:	A *pdf* file ('hormPlot') with all graphs

Code

```
#Basic code#
  hormPlot(x = base)

#Full code#
  hormPlot(x = base, date_format = 'MMM-dd', xscale = 'fixed',
  yscale = 'fixed', plot_per_page = 3, plot_height = 3, plot_width = 5,
  filename = 'LynxPlots', save_plot = FALSE)
```

Instructions

1. Do *not* need to name a new object
2. Call the *hormPlot()* function
3. Specify function arguments in parentheses

   a. **x** – name of the hormLong object created in *hormBaseline*
   Required

   b. **date_format** – format of the date on the x-axis.
   Default set to '%d-%b' (e.g. 22-Jun)
   See Appendix 1 for more information about date formats.

   c. **xscale** – whether the scale of the x-axis is 'free' or 'fixed'
   Default set to 'free'
   If set to 'free', then the x-axis scale will be free to change for each graph and will be automatically determined based on the data range. If set to 'fixed', then the range of the x-axis scale will be the same for all graphs.

   d. **yscale** – whether the scale of the y-axis is 'free' or 'fixed'

Default set to 'free'
If set to 'free', then the y-axis scale will be free to change for each graph and will be automatically determined based on the data range. If set to 'fixed', then the range of the y-axis scale will be the same for all graphs.

e. **plot_per_page** – number of graphs per page
Default set to 4

f. **plot_height** – height of each graph in inches
Default set to 2
The height of the *pdf* page is automatically determined based on the both plot_per_page and plot_height.

g. **plot_width** – width of each graph in inches
Default set to 6
Width of graphs is the same as the width of the *pdf* page.

h. **filename** – name of the *pdf* file
Default set to 'hormPlot'

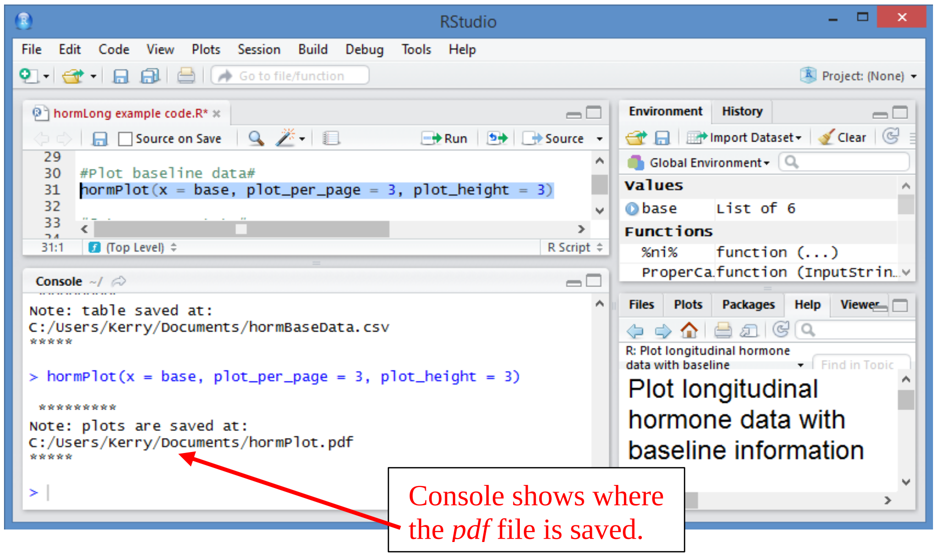
i. **save_plot** – whether to save graphs as a *pdf* file
Default set to TRUE

j. '**...**' represents other graphing options
Optional
Use help (F1) to learn more about general graphing options available in R

<u>Example</u>

DATA: hormLynx

CODE:

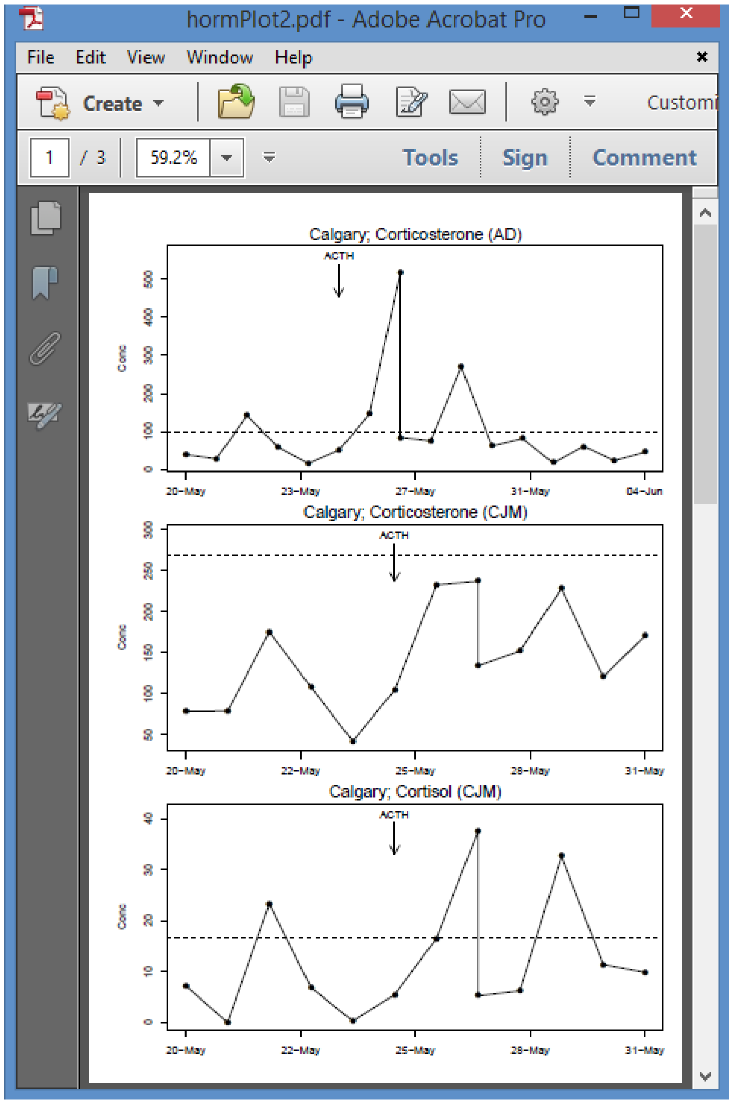


```
#Plot baseline data#
hormPlot(x = base, plot_per_page = 3, plot_height = 3)
```

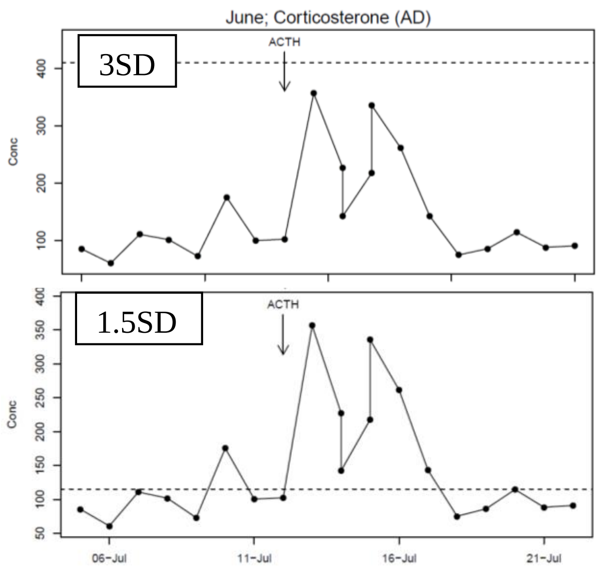
```
Values
base        List of 6
Functions
%ni%        function (...)
ProperCa    function (InputStrin…
```

```
Note: table saved at:
C:/Users/Kerry/Documents/hormBaseData.csv
*****

> hormPlot(x = base, plot_per_page = 3, plot_height = 3)

 *********
Note: plots are saved at:
C:/Users/Kerry/Documents/hormPlot.pdf
*****

> |
```

Console shows where the *pdf* file is saved.



Comparison of 3 different antibodies for one individual. The first antibody detects a nice strong peak, but the second antibody does not detect any peaks after the ACTH injection.



Comparison of different criteria for calculating baseline (top = 3SD; bottom = 1.5SD). As you increase the number of standard deviations used, you decrease the sensitivity to peaks.

hormPlotBreaks() **–** longitudinal plots with gaps

Purpose

hormPlotBreaks() is similar to *hormPlot()*, but allows you to insert breaks in the x-axis. This function is useful if you have large gaps in data collection.

Overview

INPUT:	Data object created by *hormBaseline()*
ACTIONS:	Create longitudinal graphs
OUTPUT:	A *pdf* file ('hormPlotBreak') with all graphs

Code

```
#Basic code#
  hormPlotBreaks(x = base, break_cutoff = 20, break_buffer = 30)

#Full code#
  See hormPlot()
```

Instructions

1. Do *not* need to name a new object
2. Call the *hormPlotBreaks()* function
3. Specify function arguments in parentheses.

    a. Same list of arguments as *hormPlot()*, plus 2 additional options:

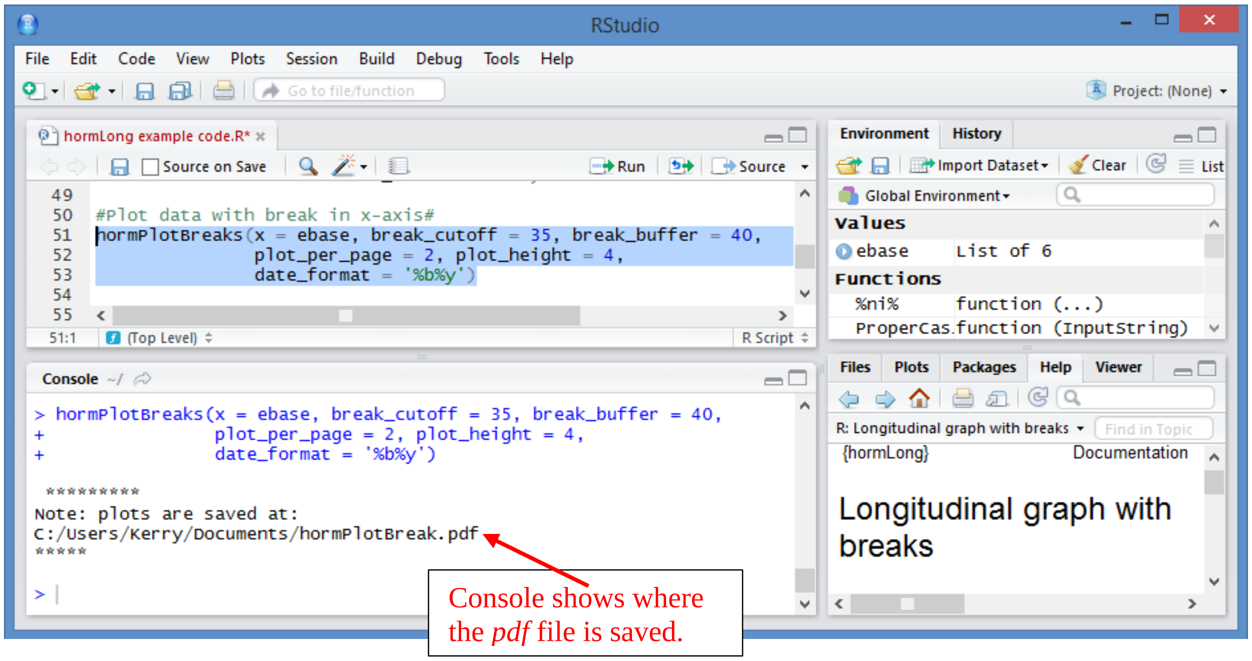
    b. **break_cutoff** – maximum number of days between consecutive data points
    Default set to 40 days
    This option allows you to specify the maximum number of days between samples that are part of the same sampling window. Intervals that are greater than this will create a break in the x-axis. If you use the default option, than any samples collected within 40 days of each other will be plotted continuously, but samples that are collected 41 days apart will create a break in the x-axis.

    c. **break_buffer** – width of break in x-axis (in number of days)
    Default set to 60 days
    This option allows you to specify the size of the gap between groups of data. If you want the data to be plotted continuously (even if there are large temporal gaps between data), then set this option to 0. As you increase the value, the gap becomes larger.

Example

DATA: hormElephant

CODE:

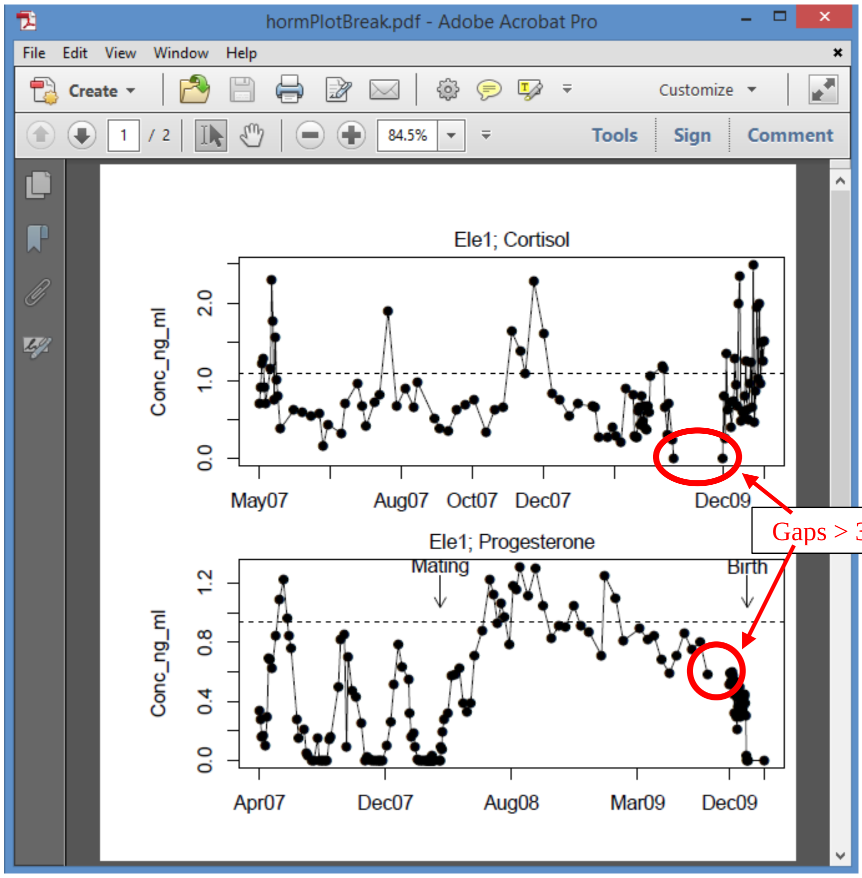


```
#Plot data with break in x-axis#
hormPlotBreaks(x = ebase, break_cutoff = 35, break_buffer = 40,
               plot_per_page = 2, plot_height = 4,
               date_format = '%b%y')
```

```
> hormPlotBreaks(x = ebase, break_cutoff = 35, break_buffer = 40,
+                plot_per_page = 2, plot_height = 4,
+                date_format = '%b%y')

 *********
Note: plots are saved at:
C:/Users/Kerry/Documents/hormPlotBreak.pdf
*****

>
```

Console shows where the *pdf* file is saved.

OUTPUT:



Example of *hormPlotBreaks()* for 2 different hormones. Cortisol data (top) was collected prior to pregnancy and for a few days at the end of the pregnancy. Progesterone data was collected much more regularly.
**Note*: if sample dates do not overlap with event dates, then events do not appear on the graph (e.g., cortisol).

hormSumTable() – Getting summary statistics

Purpose

hormSumTable() is used to calculate summary statistics for hormone data. Stats are calculated separately for each group identified by the *by_var* statement in *hormBaseline()* (e.g., different individuals, hormones, etc).

Overview

INPUT:	Data object created by *hormBaseline()*
ACTIONS:	Calculate summary statistics
OUTPUT:	A *csv* file ('hormSumTable') with all summary statistics

Code

```
#Basic code#
  hormSumTable(x = base)

#Full code#
  hormSumTable(x = base, num_decimals = 3)
```

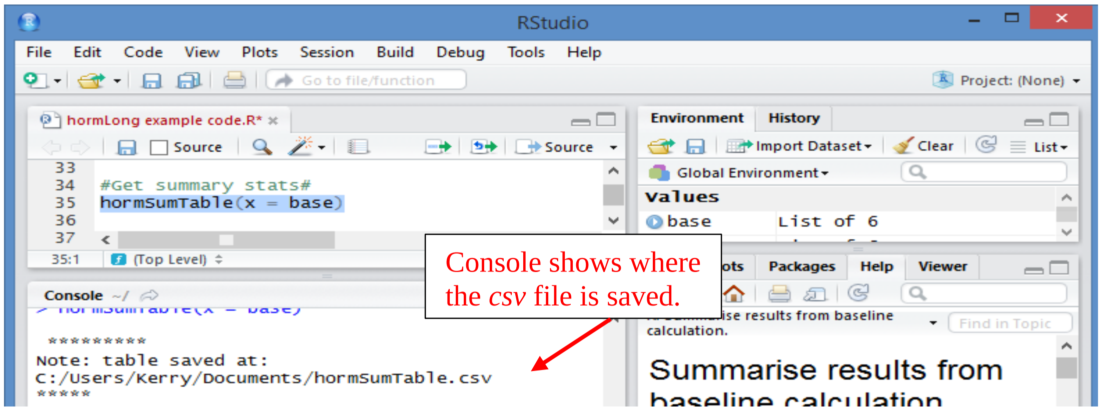
Instructions

1. Do *not* need to name a new object
2. Call the *hormSumTable()* function
3. Specify function arguments in parentheses
   a. **x** – name of the hormLong object created in *hormBaseline*
      Required
   b. **num_decimals** – number of decimal places shown
      Default set to 2

Example

DATA: hormLynx

CODE:



```
#Get summary stats#
hormSumTable(x = base)
```

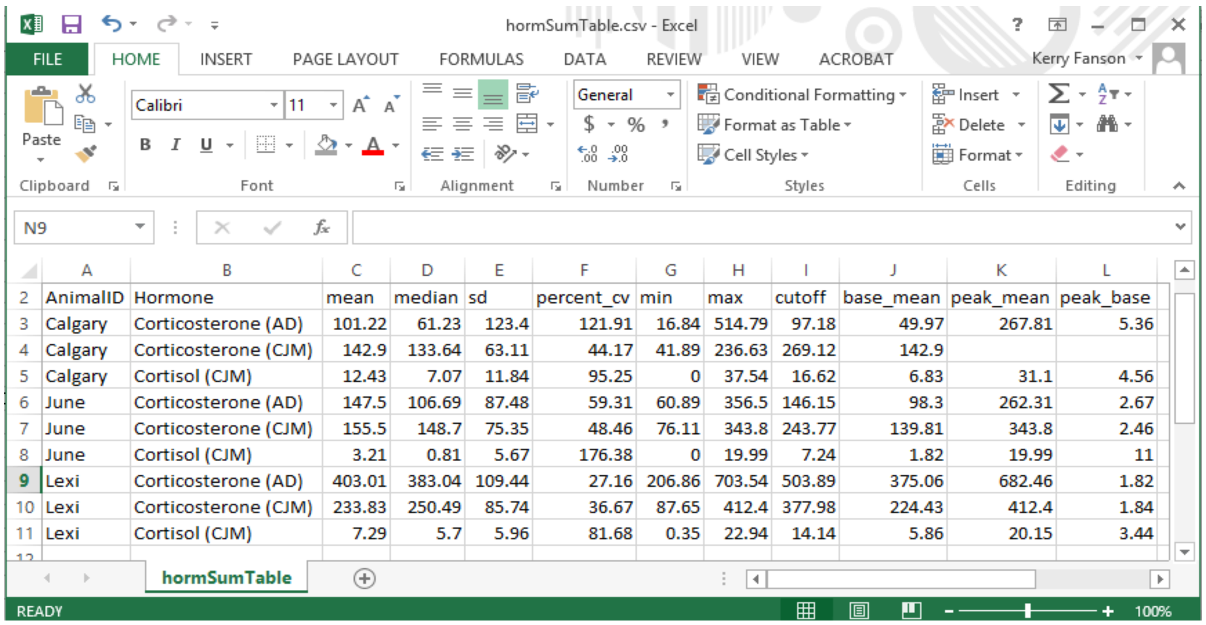
```
Values
base          List of 6
```

```
 *********
Note: table saved at:
C:/Users/Kerry/Documents/hormSumTable.csv
*****
```

ise results from baseline calculation.

Summarise results from baseline calculation

OUTPUT:



	A	B	C	D	E	F	G	H	I	J	K	L
2	AnimalID	Hormone	mean	median	sd	percent_cv	min	max	cutoff	base_mean	peak_mean	peak_base
3	Calgary	Corticosterone (AD)	101.22	61.23	123.4	121.91	16.84	514.79	97.18	49.97	267.81	5.36
4	Calgary	Corticosterone (CJM)	142.9	133.64	63.11	44.17	41.89	236.63	269.12	142.9		
5	Calgary	Cortisol (CJM)	12.43	7.07	11.84	95.25	0	37.54	16.62	6.83	31.1	4.56
6	June	Corticosterone (AD)	147.5	106.69	87.48	59.31	60.89	356.5	146.15	98.3	262.31	2.67
7	June	Corticosterone (CJM)	155.5	148.7	75.35	48.46	76.11	343.8	243.77	139.81	343.8	2.46
8	June	Cortisol (CJM)	3.21	0.81	5.67	176.38	0	19.99	7.24	1.82	19.99	11
9	Lexi	Corticosterone (AD)	403.01	383.04	109.44	27.16	206.86	703.54	503.89	375.06	682.46	1.82
10	Lexi	Corticosterone (CJM)	233.83	250.49	85.74	36.67	87.65	412.4	377.98	224.43	412.4	1.84
11	Lexi	Cortisol (CJM)	7.29	5.7	5.96	81.68	0.35	22.94	14.14	5.86	20.15	3.44

Description of Table Statistics

Mean	average (of all points for that set of grouping variables)
median	median
sd	standard deviation
percent_cv	percent coefficient of variation (SD/mean*100)
min, max	Minimum and maximum values
cutoff	threshold value for peaks, calculated as mean+*n**SD for final iteration of baseline calculation (i.e., when no more points are removed). Points below this are baseline and above are peaks.
base_mean	average of all points classified as baseline
peak_mean	average of all points classified as peaks
peak_base	ratio of baseline-to-peak (calculated as peak_mean/base_mean)

hormArea() – Calculate area under the curve (AUC)

Purpose

hormArea() is used to calculate and graph the area under the curve. This is particularly useful for measuring response to an ACTH or GnRH challenge. AUC is calculated using the trapezoid method, and there are several options for what is included in the calculation: (1) integrated response – includes the entire area under the curve, with the lower-bound being the origin of the y-axis; (2) corrected integrated response – only includes the area above baseline, with the lower-bound being the mean baseline value; (3) peak response – only includes area above the threshold cutoff for baseline, with the lower-bound being the cutoff.

Overview

INPUT:	Data object created by *hormBaseline()*
ACTIONS:	Calculate area under the curve
OUTPUT:	A *csv* file with area of each peak A *pdf* file ('hormArea') with all graphs

Code

```
#Basic code#
  hormArea(x = base)

#Full code#
  hormArea(x = base, lower_bound = 'peak', method = 'trapezoid',
  plot_per_page = 3, plot_height = 3, plot_width = 5, yscale = 'fixed',
  xscale = 'fixed', save_plot = FALSE)
```

Instructions

2. Do *not* need to name a new object
3. Call the *hormArea()* function
4. Specify function arguments in parentheses

   a. **x** – name of the hormLong object created in *hormBaseline*
   Required

   b. **lower_bound** – lower limit of y-axis for AUC calculation
   Default set to 'origin'
   Determines lower boundary of y-axis for calculating AUC. Options are: 'origin' – y=0; 'baseline' – y=mean baseline; or 'peak' – y=cutoff

   c. **method** – calculation method for AUC
   Default set to 'trapezoid'

   d. **date_format** – format of the date on the x-axis.

Default set to '%d-%b' (e.g. 22-Jun)

e. **xscale** – whether the scale of the x-axis is 'free' or 'fixed'
Default set to 'free'

f. **yscale** – whether the scale of the y-axis is 'free' or 'fixed'
Default set to 'free'

g. **plot_per_page** – number of graphs per page
Default set to 4

h. **plot_height** – height of each graph in inches
Default set to 2

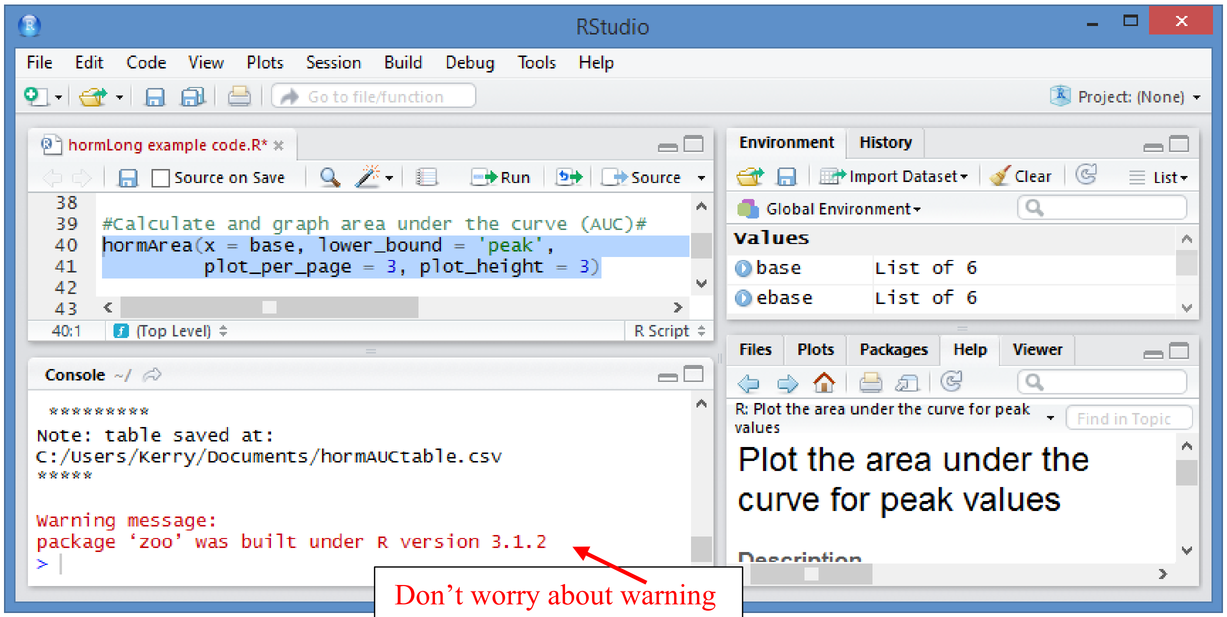
i. **plot_width** – width of each graph in inches
Default set to 6

j. **save_plot** – whether to save graphs as a *pdf* file
Default set to TRUE

Example

DATA: hormLynx

CODE:



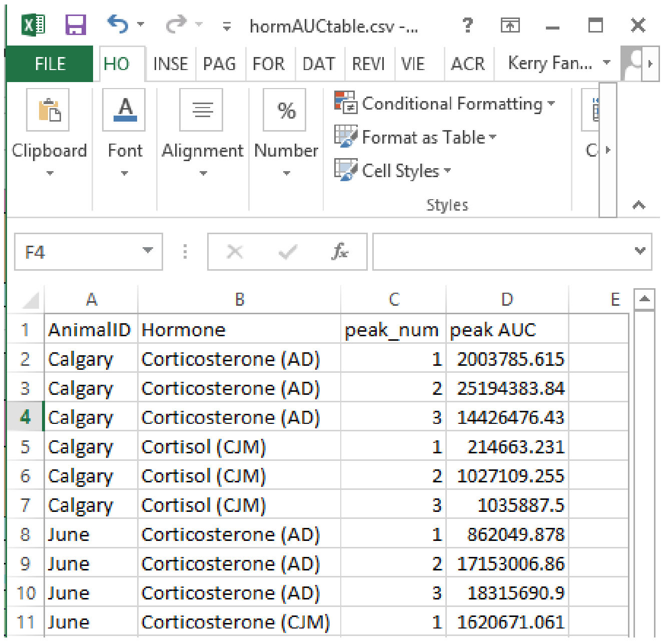
```
#Calculate and graph area under the curve (AUC)#
hormArea(x = base, lower_bound = 'peak',
         plot_per_page = 3, plot_height = 3)
```

```
Values
base     List of 6
ebase    List of 6
```

```
*********
Note: table saved at:
C:/Users/Kerry/Documents/hormAUCtable.csv
*****

Warning message:
package 'zoo' was built under R version 3.1.2
>
```

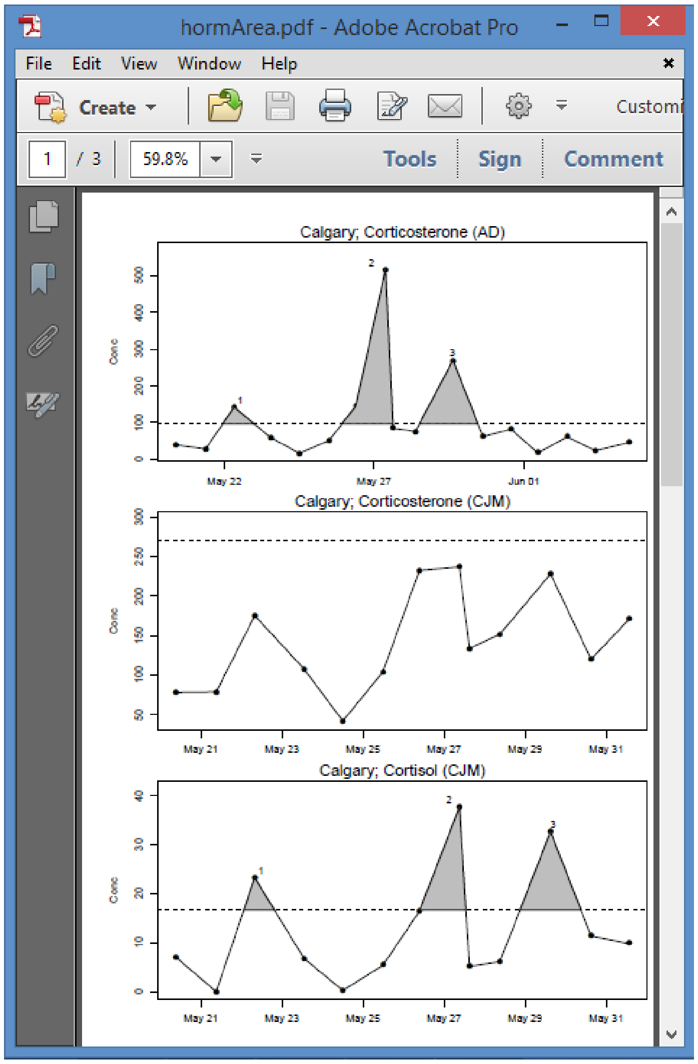
Don't worry about warning message for 'zoo'

OUTPUT:

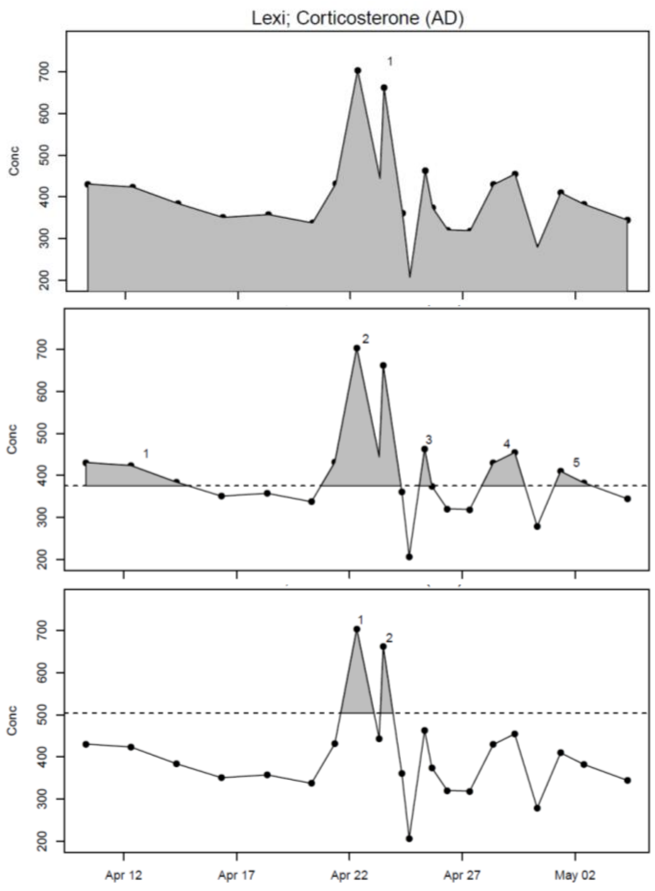


	A	B	C	D
1	AnimalID	Hormone	peak_num	peak AUC
2	Calgary	Corticosterone (AD)	1	2003785.615
3	Calgary	Corticosterone (AD)	2	25194383.84
4	Calgary	Corticosterone (AD)	3	14426476.43
5	Calgary	Cortisol (CJM)	1	214663.231
6	Calgary	Cortisol (CJM)	2	1027109.255
7	Calgary	Cortisol (CJM)	3	1035887.5
8	June	Corticosterone (AD)	1	862049.878
9	June	Corticosterone (AD)	2	17153006.86
10	June	Corticosterone (AD)	3	18315690.9
11	June	Corticosterone (CJM)	1	1620671.061

Exported *csv* file showing the AUC for each peak. In this case, peaks were only calculated as area above the baseline cutoff.



Exported *pdf* file showing the AUC for each peak (shaded area). In this case, peaks were only calculated as area above the baseline cutoff.



Comparison of different lower-bound approaches for AUC calculation:
top = ‘origin’;
middle = ‘baseline’;
bottom = ‘peak’

hormBoxPlot() – Comparing individuals

Purpose

hormBoxPlot() creates a graph consisting of vertical boxplots for each individual. Separate graphs can be created using the *by_var* option to specify grouping factors (e.g., hormone, sex). This function is useful for visualizing the patterns in mean and variance among individuals.

Overview

INPUT:	Dataset originally imported into R
ACTIONS:	Create boxpots for each individual
OUTPUT:	A *pdf* file (‘hormBoxplot’) with all graphs

Code

```
#Basic code#
  hormBoxPlot(data = ds, conc_var = ‘Conc’, id_var = ‘AnimalID’)

#Full code#
  hormBoxPlot(data = ds, conc_var = ‘Conc’, id_var = ‘AnimalID’,
  by_var = ‘Hormone’, plot_height = 3, plot_width = 5, save_plot = FALSE,
  log_scale = ‘y’)
```

Instructions

1. Do *not* need to name a new object
2. Call the *hormBoxPlot()* function
3. Specify function arguments in parentheses

    a. **data** – name of the dataset
    Required

    b. **conc_var** – name of concentration variable
    Required

    c. **id_var** – name of animal ID column
    Required

    d. **by_var** – name of grouping variables
    Optional
    If you want to create separate graphs for different grouping variables (e.g., hormone, sex) then specify that here.

    e. **log_scale** – whether y-axis is in $\log_{10}$ scale
    Default set to ‘n’ (no)

If you created a log-transformed variable in your dataset, then either call that variable in the 'conc_var' option ***or*** set 'log_scale' = 'y'. Do not use both options.

f. **plot_per_page** – number of graphs per page
Default set to 4

g. **plot_height** – height of each graph in inches
Default set to 2

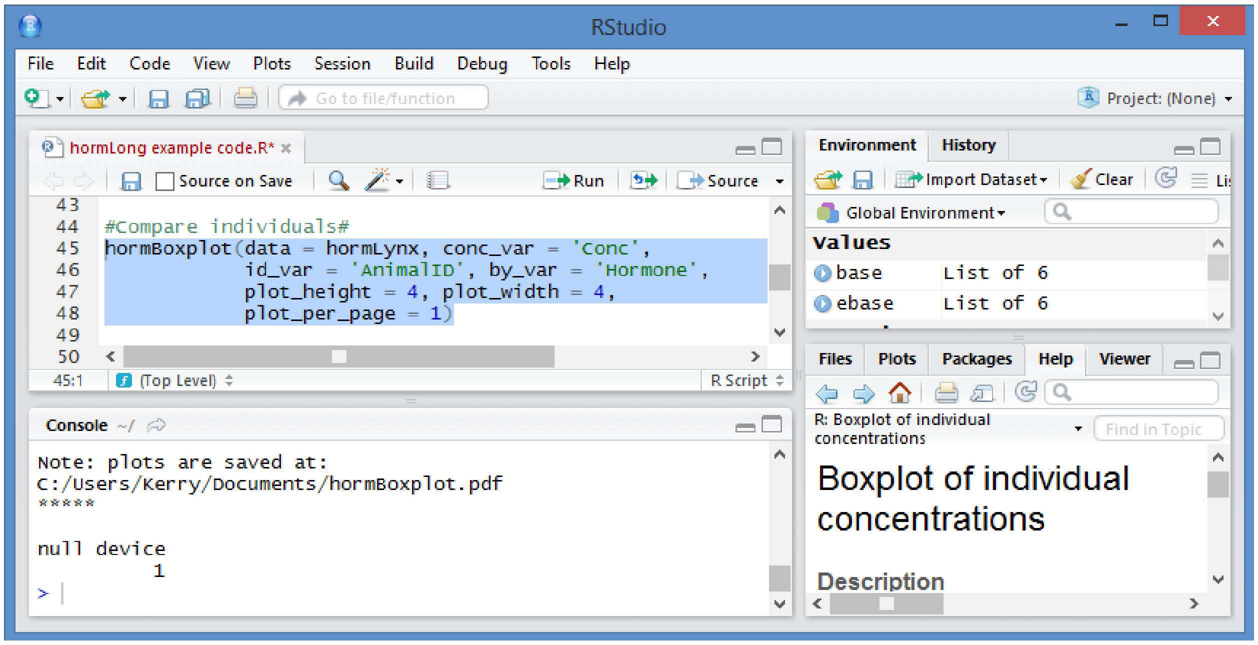
h. **plot_width** – width of each graph in inches
Default set to 6

i. **save_plot** – whether to save graphs as a *pdf* file
Default set to TRUE

Example

DATA: hormLynx

CODE:

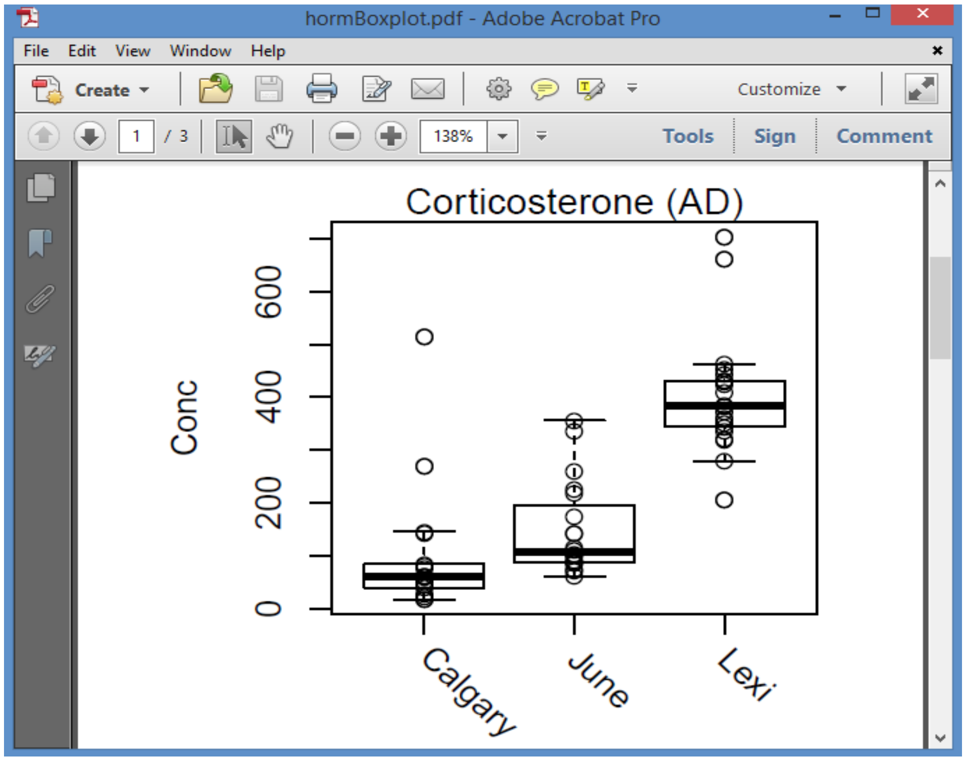


```
#Compare individuals#
hormBoxplot(data = hormLynx, conc_var = 'Conc',
            id_var = 'AnimalID', by_var = 'Hormone',
            plot_height = 4, plot_width = 4,
            plot_per_page = 1)
```

```
Note: plots are saved at:
C:/Users/Kerry/Documents/hormBoxplot.pdf
*****

null device
          1
>
```

```
Values
base     List of 6
ebase    List of 6
```

OUTPUT:


hormBoxplot.pdf - Adobe Acrobat Pro
File Edit View Window Help
Create
Customize
1 / 3
138%
Tools Sign Comment
Corticosterone (AD)
Conc
0
200
400
600
Calgary
June
Lexi

hormPlotOverlap() – Plotting multiple hormones

Purpose

hormPlotOverlap() is used to plot multiple hormones on the same graph.

Overview

INPUT:	Data object created by *hormBaseline()*
ACTIONS:	Create longitudinal graphs with multiple hormones
OUTPUT:	A *pdf* file ('hormPlotOverlap') with all graphs

Code

```
#Basic code#
  hormPlotOverlap(x = base, hormone_var = 'Hormone')

#Full code#
  hormPlotOverlap(x = base, hormone_var = 'Hormone',
  date_format = '%b%y', colors = 'yellow, green, purple', plot_per_page = 3,
  plot_height = 3, plot_width = 5, yscale = 'fixed', xscale = 'fixed',
  save_plot = FALSE)
```

Instructions

4. Do *not* need to name a new object
5. Call the *hormPlotOverlap()* function
6. Specify function arguments in parentheses

   a. **x** – name of the hormLong object created in *hormBaseline*
      Required

   b. **hormone_var** – name of column that specifies hormone type
      Required
      This **must** be included as a 'by_var' in the *hormBaseline()* statement

   c. **colors** – specify color of hormone fill
      Default set to 'red,blue'
      If you have more than 2 hormones, then you need to specify additional colors.

   d. **add_fill** – whether area under the curve should be shaded
      Default set to TRUE

   e. **two_axes** – whether to include a second y-axis
      Default set to FALSE
      If you are plotting two hormones that scale very differently, it can be useful to show the scale on separate y-axes. Set value to '*TRUE*' if you want to have 2 y-axes. **Note*: This

option can only be used with 2 hormones. If there are more than 2 hormones then an error message will appear.

f. **date_format** – format of the date on the x-axis.
Default set to '%d-%b' (e.g. 22-Jun)

g. **xscale** – whether the scale of the x-axis is 'free' or 'fixed'
Default set to 'free'

h. **yscale** – whether the scale of the y-axis is 'free' or 'fixed'
Default set to 'free'

i. **plot_per_page** – number of graphs per page
Default set to 4

j. **plot_height** – height of each graph in inches
Default set to 2

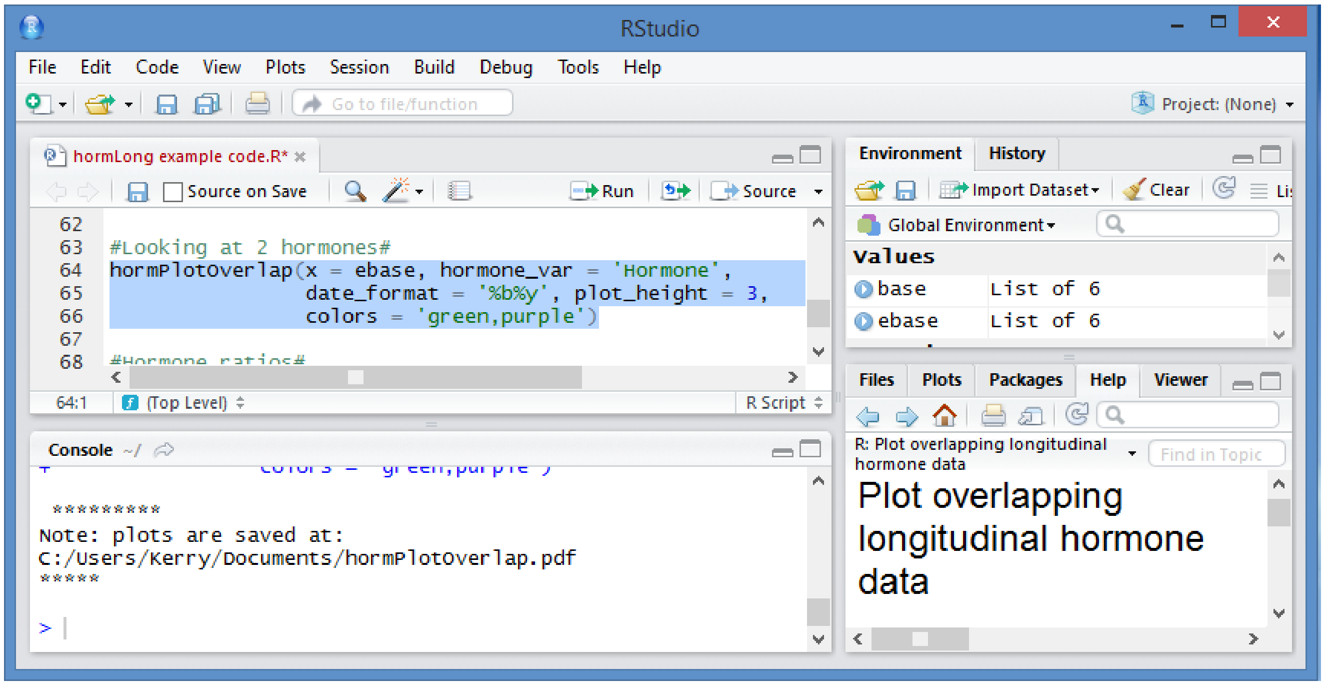
k. **plot_width** – width of each graph in inches
Default set to 6

l. **save_plot** – whether to save graphs as a *pdf* file
Default set to TRUE

Example

DATA: hormElephant

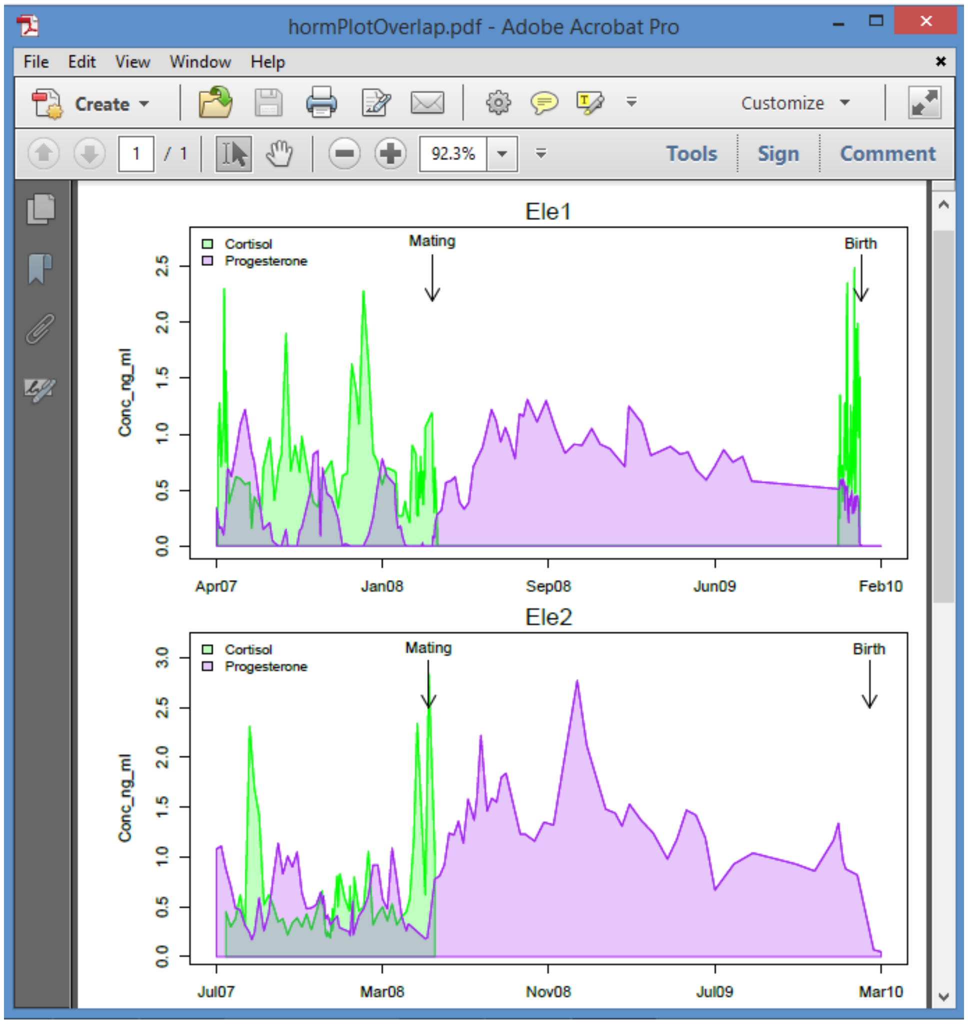
CODE:



```
#Looking at 2 hormones#
hormPlotOverlap(x = ebase, hormone_var = 'Hormone',
                date_format = '%b%y', plot_height = 3,
                colors = 'green,purple')

#Hormone ratios#
```

```
+                colors = 'green,purple')

*********
Note: plots are saved at:
C:/Users/Kerry/Documents/hormPlotOverlap.pdf
*****

>
```

OUTPUT:



Example of *hormPlotOverlap()*. Shading helps highlight the inter-luteal increases in cortisol.

Note: If you have gaps in 1 dataset but not the other (e.g., Ele1 above), setting the concentration to zero at the beginning and end of the gap will help create cleaner plots.

hormPlotRatio() – Ratio between 2 hormones

Purpose

hormPlotRatio() is used to look at the ratio between 2 hormones over time. You can specify which hormone is the numerator and which is the denominator.

Overview

INPUT:	Data object created by *hormBaseline()*
ACTIONS:	Create longitudinal graphs of ratios
OUTPUT:	A *pdf* file (‘hormPlotRatio’) with all graphs

Code

```
#Code#
  hormPlotRatio(x = base, hormone_var = 'Hormone',
  hormone_num = 'Progesterone', hormone_denom = 'Cortisol')
```

Instructions

1. Do *not* need to name a new object
2. Call the *hormPlotRatio()* function
3. Specify function arguments in parentheses

    a. **x** – name of the hormLong object created in *hormBaseline*
    Required

    b. **hormone_var** – name of column that specifies hormone type
    Required
    This **must** be included as a ‘by_var’ in the *hormBaseline()* statement

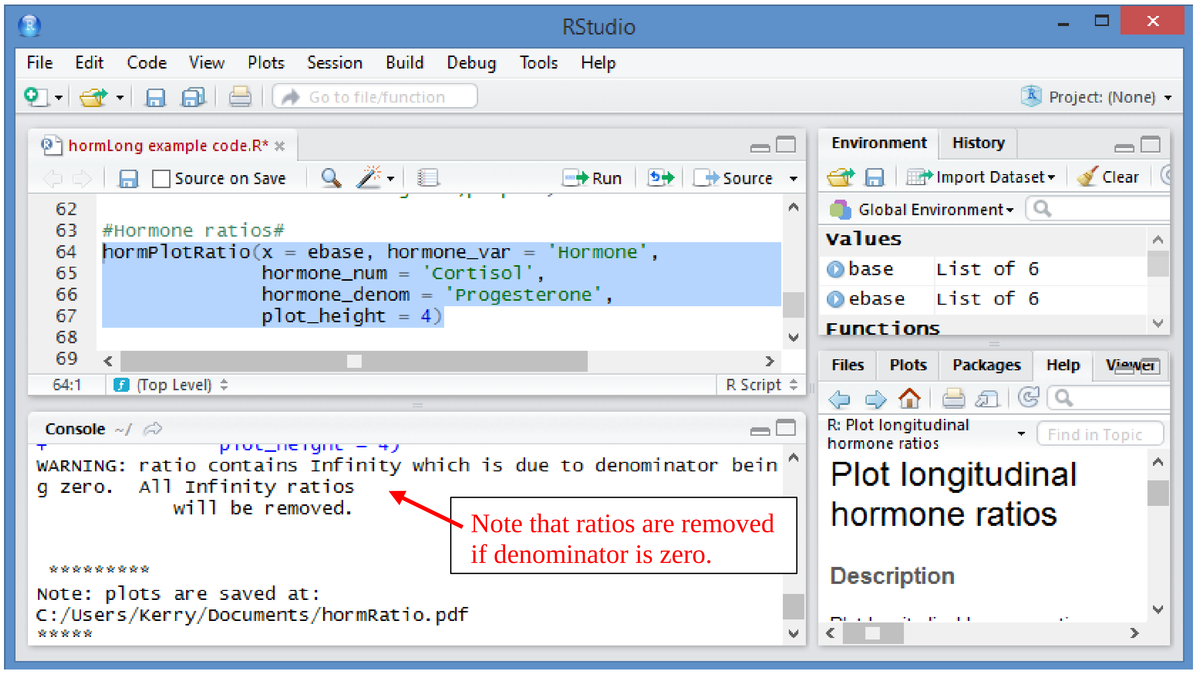
    c. **hormone_num** – name of hormone that should be the numerator
    Required

    d. **hormone_denom** – name of hormone that should be the denominator
    Required

Example

DATA: hormElephant

CODE:

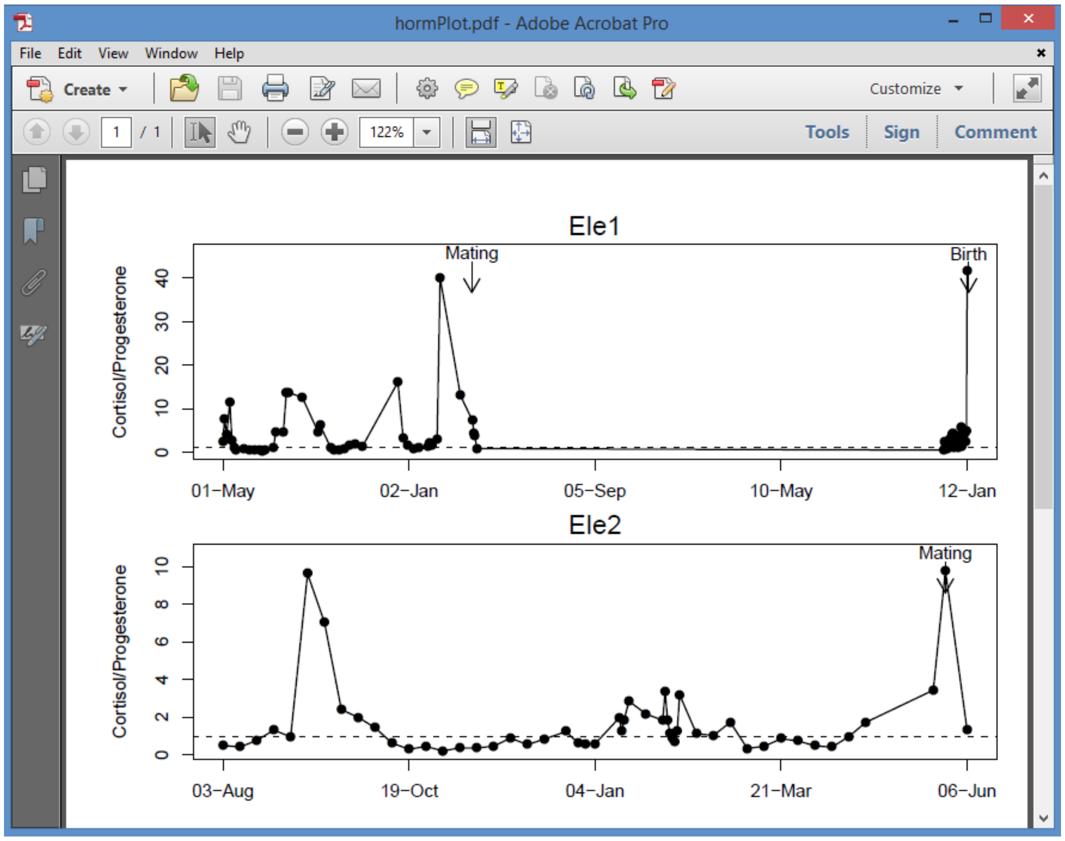


```
#Hormone ratios#
hormPlotRatio(x = ebase, hormone_var = 'Hormone',
              hormone_num = 'Cortisol',
              hormone_denom = 'Progesterone',
              plot_height = 4)
```

```
WARNING: ratio contains Infinity which is due to denominator bein
g zero.  All Infinity ratios
    will be removed.


 *********
Note: plots are saved at:
C:/Users/Kerry/Documents/hormRatio.pdf
*****
```

Note that ratios are removed if denominator is zero.

OUTPUT:



Appendix 1: Table of date formats for date format

Examples of common date_formats

Code	Description	Example
‘%d-%m-%y’	Day-Month-ShortYear (all numeric)	’22-10-14’
‘%d-%b’	Day-AbbreviatedMonth	’22-Oct’
‘%d/%b’	Day/AbbreviatedMonth	‘22/Oct’
‘%Y%b%d’	YearAbbreviatedMonthDay	‘2014Oct22’
‘%Y%b’	YearAbbreviatedMonth	‘2014Oct’
‘%a’	DayOfWeek	‘Mon’
‘%d%B%Y’	DayMonthYear	‘22October2014’
‘%d %B, %Y’	Day Month, Year	’22 October, 2014’

List of all codes for various date / time formats

Code	Description
%a	Abbreviated weekday name in the current locale. (Also matches full name on input.)
%A	Full weekday name in the current locale. (Also matches abbreviated name on input.)
%b	Abbreviated month name in the current locale. (Also matches full name on input.)
%B	Full month name in the current locale. (Also matches abbreviated name on input.)
%c	Date and time. Locale-specific on output, "%a %b %e %H:%M:%S %Y" on input.
%C	Century (00–99): the integer part of the year divided by 100.
%d	Day of the month as decimal number (01–31).
%D	Date format such as %m/%d/%y: ISO C99 says it should be that exact format.
%e	Day of the month as decimal number (1–31), with a leading space for a single-digit number.
%F	Equivalent to %Y-%m-%d (the ISO 8601 date format).
%g	The last two digits of the week-based year (see %V). (Accepted but ignored on input.)
%G	The week-based year (see %V) as a decimal number. (Accepted but ignored on input.)
%h	Equivalent to %b.
%H	Hours as decimal number (00–23). As a special exception strings such as 24:00:00 are accepted for input, since ISO 8601 allows these.
%I	Hours as decimal number (01–12).
%j	Day of year as decimal number (001–366).
%m	Month as decimal number (01–12).
%M	Minute as decimal number (00–59).
%n	Newline on output, arbitrary whitespace on input.
%p	AM/PM indicator in the locale. Used in conjunction with %I and **not**with %H. An empty string in some locales (and the behaviour is undefined if used for input in such a locale).
%r	Some platforms accept %P for output, which uses a lower-case version: others will output P.
%R	The 12-hour clock time (using the locale's AM or PM). Only defined in some locales.
%S	Equivalent to %H:%M.
%t	Second as decimal number (00–61), allowing for up to two leap-seconds (but POSIX-compliant implementations will ignore leap seconds).
%T	Tab on output, arbitrary whitespace on input.
%u	Equivalent to %H:%M:%S.
%U	Weekday as a decimal number (1–7, Monday is 1).
%V	Week of the year as decimal number (00–53) using Sunday as the first day 1 of the week (and typically with the first Sunday of the year as day 1 of week 1). The US convention.

%w	Week of the year as decimal number (00–53) as defined in ISO 8601. If the week (starting on Monday) containing 1 January has four or more days in the new year, then it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. (Accepted but ignored on input.)
%W	Weekday as decimal number (0–6, Sunday is 0).
%x	Week of the year as decimal number (00–53) using Monday as the first day of week (and typically with the first Monday of the year as day 1 of week 1). The UK convention.
%X	Date. Locale-specific on output, "%y/%m/%d" on input.
%y	Time. Locale-specific on output, "%H:%M:%S" on input.
%Y	Year without century (00–99). On input, values 00 to 68 are prefixed by 20 and 69 to 99 by 19 – that is the behaviour specified by the 2004 and 2008 POSIX standards, but they do also say ‘it is expected that in a future version the default century inferred from a 2-digit year will change’.

References

Brown, J. L., D. E. Wildt, N. Wielebnowski, K. L. Goodrowe, L. H. Graham, S. Wells, and J. G. Howard. 1996. Reproductive activity in captive female cheetahs (*Acinoyx jubatus*) assessed by faecal steroids. Journal of Reproduction and Fertility **106**:337-346.

Clifton, D. K., and R. A. Steiner. 1983. Cycle Detection: A Technique for Estimating the Frequency and Amplitude of Episodic Fluctuations inBlood Hormone and Substrate Concentrations. Endocrinology **112**:1057-1064.

Fanson, K. V., T. Keeley, and B. G. Fanson. 2014. Cyclic changes in cortisol across the estrous cycle in parous and nulliparous Asian elephants. Endocrine connections **3**:57-66.

Fanson, K. V., N. C. Wielebnowski, T. M. Shenk, and J. R. Lucas. 2012. Comparative patterns of adrenal activity in captive and wild Canada lynx (*Lynx canadensis*). Journal of Comparative Physiology B: Biochemical, Systemic, and Environmental Physiology **182**:157-165.