

Guía 2: Estructuras de datos

Bioinformática - Pasto 2017

Guillermo Torres

Variables

Las variables son el elemento base en programación. En una variable se almacena un valor y/o información. Esta información pueden ser otras estructuras de datos. Una variable necesita ser definida; tener un nombre y un valor el cual se asigna con el operador asignación `<-` o `=` aunque se recomienda usar el operador asignación, para evitar confusiones.

```
peso <- 45
alturas <- c(120,132,145)
nombre <- "Guillermo"
generacion <- factor(c(rep("joven",2),rep("adultos",3),rep("viejos",5)),levels=c("joven","adultos","viejos"))
```

Para recuperar los valores:

```
peso
```

```
## [1] 45
```

```
alturas
```

```
## [1] 120 132 145
```

```
nombre
```

```
## [1] "Guillermo"
```

```
generacion
```

```
## [1] joven  joven  adultos adultos adultos viejos  viejos  viejos
```

```
## [9] viejos  viejos
```

```
## Levels: joven adultos viejos
```

Clases de las variables

Cada variable puede ser de diferentes tipos. Numérica (numeric), caracteres (character), categóricos (factor), listas (list), data frames o matrices.

```
class(peso)
```

```
## [1] "numeric"
```

```
class(alturas)
```

```
## [1] "numeric"
```

```
class(nombre)
```

```
## [1] "character"
```

```
class(generacion)
```

```
## [1] "factor"
```

Vectores

Los vectores son listas de elementos que no están indexados. Los elementos que ahí están contenidos pueden ser numéricos, caracteres o factores. Para asignar un vector se usa la función `c()`:

```
alturas <- c(120,132,145)
```

Listas

A diferencia de los vectores, esta estructura de datos es una lista de elementos indexada. Los elementos ahí indexados pueden ser numéricos, caracteres, factores e incluso dataframes o matrices y más. Para crear una lista se utiliza la función `list()`:

```
individuos <- list(nombres=c("Guillermo", "Milena"), alturas=c(180,170), genero=c("hombre", "mujer"))
```

Para explorar los contenidos de las listas se usa el operador `$` seguido del nombre del índice:

```
individuos$nombres
```

```
## [1] "Guillermo" "Milena"
```

```
class(individuos$nombres)
```

```
## [1] "character"
```

Data frames y Matrices

Estas estructuras de datos son las más similares a una tabla de Excel. Se caracteriza por tener filas y columnas y almacenar información de una forma matricial. La diferencia entre data frames y matrices es que los data frames pueden tener columnas cuya clase sea diferente, mientras que las matrices solo aceptan columnas de la misma clase.

```
datos <- data.frame(nombres=c("Guillermo", "Milena"), alturas=c(180,170), genero=c("hombre", "mujer"))
datos.matriz <- matrix(1:6, nrow=2, ncol=3)
```

Para explorar los contenidos de dataframes se usa el operador `$` seguido del nombre de la columna. Cuando se inserta información en un dataframe, la función dataframe corre por defecto un proceso que se llama coerción. La coerción recodifica la clase de cada columna:

```
datos$nombres
```

```
## [1] Guillermo Milena
```

```
## Levels: Guillermo Milena
```

```
class(datos$nombres)
```

```
## [1] "factor"
```

What is dplyr?

`dplyr` es un poderoso paquete de R que permite transformar y resumir datos tabulares con filas y columnas. Para una explicación más profunda revisar las vignettes del paquete: [Introduction to dplyr](#)

Por que es útil?

El paquete contiene un conjunto de funciones (o “verbs”) que permiten manipular los datos con mayor simplicidad: por ejemplo, filtrar por filas, seleccionar columnas específicas, re-ordenar filas, añadir columnas y resumir los datos. Adicionalmente, dplyr contiene funciones útiles para desarrollar tareas comunes las cuales se clasifican bajo el concepto “split-apply-combine”.

Como se realizan comparaciones usando las funciones base de R?

Al trabajar con R, probablemente uno llegue a familiarizarse con funciones tales como `split()`, `subset()`, `apply()`, `sapply()`, `lapply()`, `tapply()` o `aggregate()`, las cuales permiten manipular los data y aplicar sobre ellos diferentes tipos de funciones. Sin embargo, las funciones en `dplyr` que se encargan de las mismas tareas son más fáciles para trabajar, son más consistentes en la sintaxis y estan optimizadas para trabajar con data frame en vez de solo vectores.

Trabajando con dplyr

Para instalar dplyr:

```
install.packages("dplyr")
```

Para cargar dplyr:

```
library(dplyr)
```

Set the datos: sueño de los mamiferos

El set de datos `msleep` (mammals sleep) contiene los tiempos de sueño y los pesos para un conjunto de mamíferos. Este dataset contiene 83 filas y 11 variables. El archivo esta en la carpeta `extdata`.

```
library(downloader)
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extdata/femaleMiceWeights.csv"
filename <- "femaleMiceWeights.csv"
download(url, destfile=filename)
msleep <- read.csv
# o
dir <- ("~/Documents/Proyectos/2017/12PastoWorkshop/CourseLab/ws/extdata/") #ruta a extdata
file <- paste0(dir,"msleep_ggplot2.csv")
msleep <- read.csv(file)
head(msleep)
```

	name	genus	vore	order	conservation	
## 1	Cheetah	Acinonyx	carni	Carnivora	lc	
## 2	Owl monkey	Aotus	omni	Primates	<NA>	
## 3	Mountain beaver	Aplodontia	herbi	Rodentia	nt	
## 4	Greater short-tailed shrew	Blarina	omni	Soricomorpha	lc	
## 5	Cow	Bos	herbi	Artiodactyla	domesticated	
## 6	Three-toed sloth	Bradypus	herbi	Pilosa	<NA>	
##	sleep_total	sleep_rem	sleep_cycle	awake	brainwt	bodywt
## 1	12.1	NA	NA	11.9	NA	50.000
## 2	17.0	1.8	NA	7.0	0.01550	0.480
## 3	14.4	2.4	NA	9.6	NA	1.350
## 4	14.9	2.3	0.1333333	9.1	0.00029	0.019

```
## 5      4.0      0.7  0.6666667 20.0 0.42300 600.000
## 6     14.4      2.2  0.7666667  9.6      NA   3.850
```

Las columnas (en orden) corresponden a:

column name	Description
name	nombre común
genus	rango taxonómico Genero
vore	carnivoro, omnivoro or herbivoro?
order	rango taxonómico Orden
conservation	status de conservación del mamifero
sleep_total	cantidad total de sueño (horas)
sleep_rem	sueño rem (horas)
sleep_cycle	longitud del ciclo de sueño (horas)
awake	cantidad de tiempo despierto (horas)
brainwt	peso del cerebro (kg)
bodywt	peso corporal (kg)

dplyr Verbs Importantes para recordar.

dplyr verbs	Description
select()	selecciona columnas
filter()	filtra filas
arrange()	re-ordena or arregla filas
mutate()	crea nuevas columnas
summarise()	resume valores
group_by()	agrupamiento bajo “split-apply-combine”

dplyr en acción

Dos funciones básicas son `select()` y `filter()`, las cuales seleccionan columnas y filtran filas respectivamente

Seleccionando columnas con `select()`

Selecconemos un par de columnas: `name` y `sleep_total`

```
sleepData <- select(msleep, name, sleep_total)
head(sleepData)
```

```
##           name sleep_total
## 1      Cheetah      12.1
## 2 Owl monkey      17.0
## 3 Mountain beaver      14.4
## 4 Greater short-tailed shrew      14.9
## 5      Cow      4.0
## 6 Three-toed sloth      14.4
```

Para seleccionar todas las columnas *excepto* una columna específica, usar el operador “-” (substracción, tambien conocio como indexación negativa):

```
head(select(msleep, -name))
```

```
##      genus vore      order conservation sleep_total sleep_rem
## 1  Acinonyx carni   Carnivora          lc         12.1        NA
## 2    Aotus omni    Primates        <NA>         17.0         1.8
## 3 Aplodontia herbi   Rodentia          nt         14.4         2.4
## 4   Blarina omni Soricomorpha          lc         14.9         2.3
## 5      Bos herbi Artiodactyla domesticated         4.0         0.7
## 6  Bradypus herbi    Pilosa        <NA>         14.4         2.2
##  sleep_cycle awake brainwt bodywt
## 1          NA   11.9      NA   50.000
## 2          NA    7.0 0.01550   0.480
## 3          NA    9.6      NA   1.350
## 4  0.1333333    9.1 0.00029   0.019
## 5  0.6666667   20.0 0.42300 600.000
## 6  0.7666667    9.6      NA   3.850
```

Para seleccionar un rango de columnas por nombre, usar el operador ":" (colon)

```
head(select(msleep, name:order))
```

```
##      name      genus vore      order
## 1   Cheetah  Acinonyx carni   Carnivora
## 2 Owl monkey    Aotus omni    Primates
## 3 Mountain beaver Aplodontia herbi   Rodentia
## 4 Greater short-tailed shrew   Blarina omni Soricomorpha
## 5      Cow      Bos herbi Artiodactyla
## 6 Three-toed sloth  Bradypus herbi    Pilosa
```

Para seleccionar todas las columnas que comienzan con las letras "sl", usar la función `starts_with()`:

```
head(select(msleep, starts_with("sl")))
```

```
##  sleep_total sleep_rem sleep_cycle
## 1         12.1         NA          NA
## 2         17.0         1.8          NA
## 3         14.4         2.4          NA
## 4         14.9         2.3  0.1333333
## 5          4.0         0.7  0.6666667
## 6         14.4         2.2  0.7666667
```

A continuación son algunas opciones adicionales para seleccionar columnas basadas en un criterio específico:

1. `ends_with()` = Selecciona columnas que terminan con sufijo específico
2. `contains()` = Selecciona columnas que contienen una palabra específico
3. `matches()` = Selecciona columnas que terminan con hagan match con una expresión regular
4. `one_of()` = Selecciona nombres de columnas que pertenecen a un grupo de nombre.

Para más información y ejemplos sobre estas funciones usar "?"

Selecting Rows Using `filter()`

Filtrar las filas por mamíferos que duermen un total de mas de 16 horas.

```
filter(msleep, sleep_total >= 16)
```

```
##      name      genus vore      order conservation
```

```
## 1 Owl monkey Aotus omni Primates <NA>
## 2 Long-nosed armadillo Dasypus carni Cingulata lc
## 3 North American Opossum Didelphis omni Didelphimorphia lc
## 4 Big brown bat Eptesicus insecti Chiroptera lc
## 5 Thick-tailed opossum Lutreolina carni Didelphimorphia lc
## 6 Little brown bat Myotis insecti Chiroptera <NA>
## 7 Giant armadillo Priodontes insecti Cingulata en
## 8 Arctic ground squirrel Spermophilus herbi Rodentia lc
## sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1 17.0 1.8 NA 7.0 0.01550 0.480
## 2 17.4 3.1 0.3833333 6.6 0.01080 3.500
## 3 18.0 4.9 0.3333333 6.0 0.00630 1.700
## 4 19.7 3.9 0.1166667 4.3 0.00030 0.023
## 5 19.4 6.6 NA 4.6 NA 0.370
## 6 19.9 2.0 0.2000000 4.1 0.00025 0.010
## 7 18.1 6.1 NA 5.9 0.08100 60.000
## 8 16.6 NA NA 7.4 0.00570 0.920
```

Filtrar las filas por mamíferos que duermen un total de mas de 16 horas *y* tienen un peso corporal de mas de 1kg.

```
filter(msleep, sleep_total >= 16, bodywt >= 1)
```

```
## name genus vore order conservation
## 1 Long-nosed armadillo Dasypus carni Cingulata lc
## 2 North American Opossum Didelphis omni Didelphimorphia lc
## 3 Giant armadillo Priodontes insecti Cingulata en
## sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1 17.4 3.1 0.3833333 6.6 0.0108 3.5
## 2 18.0 4.9 0.3333333 6.0 0.0063 1.7
## 3 18.1 6.1 NA 5.9 0.0810 60.0
```

Filtrar las filas por mamíferos que pertenecsn al orden Perissodactyla y Primates taxonomic.

```
filter(msleep, order %in% c("Perissodactyla", "Primates"))
```

```
## name genus vore order conservation
## 1 Owl monkey Aotus omni Primates <NA>
## 2 Grivet Cercopithecus omni Primates lc
## 3 Horse Equus herbi Perissodactyla domesticated
## 4 Donkey Equus herbi Perissodactyla domesticated
## 5 Patas monkey Erythrocebus omni Primates lc
## 6 Galago Galago omni Primates <NA>
## 7 Human Homo omni Primates <NA>
## 8 Mongoose lemur Lemur herbi Primates vu
## 9 Macaque Macaca omni Primates <NA>
## 10 Slow loris Nyctibeus carni Primates <NA>
## 11 Chimpanzee Pan omni Primates <NA>
## 12 Baboon Papio omni Primates <NA>
## 13 Potto Perodicticus omni Primates lc
## 14 Squirrel monkey Saimiri omni Primates <NA>
## 15 Brazilian tapir Tapirus herbi Perissodactyla vu
## sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1 17.0 1.8 NA 7.0 0.0155 0.480
## 2 10.0 0.7 NA 14.0 NA 4.750
## 3 2.9 0.6 1.0000000 21.1 0.6550 521.000
```

```
## 4      3.1      0.4      NA 20.9 0.4190 187.000
## 5     10.9      1.1      NA 13.1 0.1150  10.000
## 6      9.8      1.1 0.5500000 14.2 0.0050   0.200
## 7      8.0      1.9 1.5000000 16.0 1.3200  62.000
## 8      9.5      0.9      NA 14.5      NA   1.670
## 9     10.1      1.2 0.7500000 13.9 0.1790   6.800
## 10     11.0      NA      NA 13.0 0.0125   1.400
## 11      9.7      1.4 1.4166667 14.3 0.4400  52.200
## 12      9.4      1.0 0.6666667 14.6 0.1800  25.235
## 13     11.0      NA      NA 13.0      NA   1.100
## 14      9.6      1.4      NA 14.4 0.0200   0.743
## 15      4.4      1.0 0.9000000 19.6 0.1690 207.501
```

Es posible usar operadores booleanos para (ej. >, <, >=, <=, !=, %in%) para crear y evaluar test logicos.

El operador Pipe: %>%

Antes de seguir avanzando, vamos a introducir el operador pipe %>%. dplyr importa este operador de otro paquete (magrittr). Este operador permite concatenar las salidas de una funcion como entrada de una siguiente función. El operador rempaza las funciones anidadas (las que se leen y ejecutan de dentro hacia afuera), por expresiones de funciones que se leen y ejecutan de izquierda a derecha. Esta estructura de concatenación será usada comunmente a lo largo del curso.

Por ejemplo:

```
head(select(msleep, name, sleep_total))
```

```
##              name sleep_total
## 1          Cheetah      12.1
## 2        Owl monkey      17.0
## 3    Mountain beaver      14.4
## 4 Greater short-tailed shrew      14.9
## 5              Cow       4.0
## 6    Three-toed sloth      14.4
```

Ahora usando el operador pipe, concatenaremos el data frame `msleep` a la función que seleccionara dos columnas (`name` y `sleep_total`) y luego se concatenan a un nuevo data frame para la función `head()`, la cual nos devolvera la cabecera del nuevo data frame.

```
msleep %>%
  select(name, sleep_total) %>%
  head
```

```
##              name sleep_total
## 1          Cheetah      12.1
## 2        Owl monkey      17.0
## 3    Mountain beaver      14.4
## 4 Greater short-tailed shrew      14.9
## 5              Cow       4.0
## 6    Three-toed sloth      14.4
```

Pronto se darán cuenta de lo útil que es el operador pipe, especialmente cuando se combinen muchas funciones.

dplyr Verbs en acción

Ahora que ya se conoce acerca del operador pipe `%>%`, éste será usado en el resto de la guía.

Ordenar O Re-ordenar filas usando `arrange()`

Para ordenar (re-ordenar) filas con base en una columna particular, por ejemplo, rango taxonómico `order`, solo hay que poner el nombre de la columna desdeada en la función `'arrange()'` y las filas se ordenaran de forma alfabética:

```
msleep %>% arrange(order) %>% head
```

```
##      name      genus vore      order conservation sleep_total sleep_rem
## 1  Tenrec    Tenrec  omni Afrosoricida      <NA>         15.6        2.3
## 2    Cow      Bos   herbi Artiodactyla domesticated      4.0        0.7
## 3 Roe deer Capreolus herbi Artiodactyla      lc         3.0         NA
## 4    Goat    Capri herbi Artiodactyla      lc         5.3        0.6
## 5 Giraffe Giraffa herbi Artiodactyla      cd         1.9        0.4
## 6  Sheep     Ovis herbi Artiodactyla domesticated      3.8        0.6
##  sleep_cycle awake brainwt bodywt
## 1          NA    8.4  0.0026   0.900
## 2  0.6666667  20.0  0.4230  600.000
## 3          NA   21.0  0.0982   14.800
## 4          NA   18.7  0.1150   33.500
## 5          NA   22.1      NA  899.995
## 6          NA   20.2  0.1750   55.500
```

Ahora seleccionaremos tres columnas, ordenaremos sus filas de acuerdo al rango taxonómico `order` y por cantidad de sueño total. Finalmente, mostraremos la cabecera del data frame final:

```
msleep %>%
  select(name, order, sleep_total) %>%
  arrange(order, sleep_total) %>%
  head
```

```
##      name      order sleep_total
## 1  Tenrec Afrosoricida      15.6
## 2 Giraffe Artiodactyla      1.9
## 3 Roe deer Artiodactyla      3.0
## 4  Sheep Artiodactyla      3.8
## 5    Cow Artiodactyla      4.0
## 6  Goat Artiodactyla      5.3
```

Igual que arriba, excepto que aquí filtraremos las filas por mamíferos que duermen 16 o mas horas en vez de mostrar la cabecera del data frame resultante (note que al no usar `head` al final se mostrará todo el data frame resultante):

```
msleep %>%
  select(name, order, sleep_total) %>%
  arrange(order, sleep_total) %>%
  filter(sleep_total >= 16)
```

```
##      name      order sleep_total
## 1 Big brown bat Chiroptera      19.7
## 2 Little brown bat Chiroptera      19.9
## 3 Long-nosed armadillo Cingulata      17.4
```



```
## 4      Giant armadillo      Cingulata      18.1
## 5 North American Opossum Didelphimorphia      18.0
## 6 Thick-tailed opossum Didelphimorphia      19.4
## 7      Owl monkey      Primates      17.0
## 8 Arctic ground squirrel      Rodentia      16.6
```

Algo complicado: igual que arriba, excepto que el ordenamiento de las filas de acuerdo a la columna `sleep_total` sera en orden descendente. Para esto, usaremos la función `desc()`:

```
msleep %>%
  select(name, order, sleep_total) %>%
  arrange(order, desc(sleep_total)) %>%
  filter(sleep_total >= 16)
```

```
##           name           order sleep_total
## 1 Little brown bat Chiroptera      19.9
## 2 Big brown bat Chiroptera      19.7
## 3 Giant armadillo Cingulata      18.1
## 4 Long-nosed armadillo Cingulata      17.4
## 5 Thick-tailed opossum Didelphimorphia      19.4
## 6 North American Opossum Didelphimorphia      18.0
## 7 Owl monkey Primates      17.0
## 8 Arctic ground squirrel Rodentia      16.6
```

Creando nuevas columnas usando `mutate()`

La función `mutate()` añadirá una nueva columna al data frame. Crear una nueva columna llamada `rem_proportion`, la cual es el radio entre la cantidad de sueño `rem` y sueño total:

```
msleep %>%
  mutate(rem_proportion = sleep_rem / sleep_total) %>%
  head
```

```
##           name      genus vore      order conservation
## 1 Cheetah Acinonyx carni Carnivora lc
## 2 Owl monkey Aotus omni Primates <NA>
## 3 Mountain beaver Aplodontia herbi Rodentia nt
## 4 Greater short-tailed shrew Blarina omni Soricomorpha lc
## 5 Cow Bos herbi Artiodactyla domesticated
## 6 Three-toed sloth Bradypus herbi Pilosa <NA>
## sleep_total sleep_rem sleep_cycle awake brainwt bodywt rem_proportion
## 1 12.1 NA NA 11.9 NA 50.000 NA
## 2 17.0 1.8 NA 7.0 0.01550 0.480 0.1058824
## 3 14.4 2.4 NA 9.6 NA 1.350 0.1666667
## 4 14.9 2.3 0.1333333 9.1 0.00029 0.019 0.1543624
## 5 4.0 0.7 0.6666667 20.0 0.42300 600.000 0.1750000
## 6 14.4 2.2 0.7666667 9.6 NA 3.850 0.1527778
```

Es posible crear mas de una columna con `mutate(columnas separadas por comas)`. Añadiremos 2 columnas esta vez, `rem_proportion` y `bodywt_grams` (peso corporal en gramos):

```
msleep %>%
  mutate(rem_proportion = sleep_rem / sleep_total,
         bodywt_grams = bodywt * 1000) %>%
  head
```

```
##           name      genus vore      order conservation
```

```
## 1          Cheetah  Acinonyx carni  Carnivora      lc
## 2          Owl monkey  Aotus  omni  Primates      <NA>
## 3          Mountain beaver  Aplodontia herbi  Rodentia      nt
## 4 Greater short-tailed shrew  Blarina  omni  Soricomorpha      lc
## 5          Cow  Bos herbi  Artiodactyla domesticated
## 6          Three-toed sloth  Bradypus herbi  Pilosa      <NA>
##  sleep_total sleep_rem sleep_cycle awake brainwt  bodywt rem_proportion
## 1          12.1          NA          NA  11.9          NA  50.000          NA
## 2          17.0          1.8          NA   7.0  0.01550   0.480   0.1058824
## 3          14.4          2.4          NA   9.6          NA   1.350   0.1666667
## 4          14.9          2.3  0.1333333   9.1  0.00029   0.019   0.1543624
## 5           4.0          0.7  0.6666667  20.0  0.42300  600.000   0.1750000
## 6          14.4          2.2  0.7666667   9.6          NA   3.850   0.1527778
##  bodywt_grams
## 1          50000
## 2           480
## 3          1350
## 4           19
## 5         600000
## 6           3850
```

Crear resúmenes de los data frames usando summarise()

La función `summarise()` creará un data frame con las estadísticas solicitadas para alguna columna dada. Por ejemplo, para computar el número promedio de horas de sueño, aplicamos la función `mean()` a la columna `sleep_total` y llamamos al valor a resumir `avg_sleep`:

```
msleep %>%
  summarise(avg_sleep = mean(sleep_total))
```

```
##   avg_sleep
## 1  10.43373
```

Existen muchas otras estadísticas de resumen que se pueden considerar, tales como: `sd()`, `min()`, `max()`, `median()`, `sum()`, `n()` (devuelve la longitud del vector), `first()` (devuelve el primer valor en el vector), `last()` (devuelve el último valor en el vector) y `n_distinct()` (devuelve el número de valores distintos que hay en el vector).

```
msleep %>%
  summarise(avg_sleep = mean(sleep_total),
            min_sleep = min(sleep_total),
            max_sleep = max(sleep_total),
            total = n())
```

```
##   avg_sleep min_sleep max_sleep total
## 1  10.43373      1.9      19.9     83
```

Agrupar operaciones usando group_by()

`group_by()` es una función muy importante en `dplyr`. Como mencionamos antes está relacionada con el concepto “split-apply-combine”. Esto es por que nosotros literalmente deseamos dividir nuestro data frame original de acuerdo a alguna variable (ej. rango tax. orden), luego aplicar una función a los data frames individuales generados en el paso anterior y por último combinar los resultados en un nuevo data frame.

Lo que haremos entonces sera: dividir el data frame `msleep` de acuerdo al rango taxonómico orden, luego generar algunas estadísticas, esperando un resumen estadístico para cada orden taxonómico:

```
msleep %>%
  group_by(order) %>%
  summarise(avg_sleep = mean(sleep_total),
            min_sleep = min(sleep_total),
            max_sleep = max(sleep_total),
            total = n())
```

```
## # A tibble: 19 x 5
```

	order	avg_sleep	min_sleep	max_sleep	total
	<fctr>	<dbl>	<dbl>	<dbl>	<int>
## 1	Afrosoricida	15.600000	15.6	15.6	1
## 2	Artiodactyla	4.516667	1.9	9.1	6
## 3	Carnivora	10.116667	3.5	15.8	12
## 4	Cetacea	4.500000	2.7	5.6	3
## 5	Chiroptera	19.800000	19.7	19.9	2
## 6	Cingulata	17.750000	17.4	18.1	2
## 7	Didelphimorphia	18.700000	18.0	19.4	2
## 8	Diprotodontia	12.400000	11.1	13.7	2
## 9	Erinaceomorpha	10.200000	10.1	10.3	2
## 10	Hyracoidea	5.666667	5.3	6.3	3
## 11	Lagomorpha	8.400000	8.4	8.4	1
## 12	Monotremata	8.600000	8.6	8.6	1
## 13	Perissodactyla	3.466667	2.9	4.4	3
## 14	Pilosa	14.400000	14.4	14.4	1
## 15	Primates	10.500000	8.0	17.0	12
## 16	Proboscidea	3.600000	3.3	3.9	2
## 17	Rodentia	12.468182	7.0	16.6	22
## 18	Scandentia	8.900000	8.9	8.9	1
## 19	Soricomorpha	11.100000	8.4	14.9	5