Herramientas en R para datos complejos

Guillermo Basulto

Marzo 20, 2015

Estructura

- readr (Leer datos)
- plyr (separar-aplicar-combinar 'SAC')
- tidyr (Reestructurar datos)
- dplyr (SAC para cuadros de datos)
- ggplot2 (Gráficas)
- ggvis (Gráficas interactivas)
- Rcpp (Llamar C++ desde R)
- lubridate (manejo de fechas)

readr

- Comentar: Github y Hadley Wickham
- Instalar directamente desde Github
- Manera amigable y rápida para leer cuadros de datos
- `https://github.com/hadley/readr`
- Functiones principales: read_delim(), read_csv(), read_tsv(), read_csv2(), read_fwf(), read_table(), read_lines(), read_file()
- Tipos de variables

readr

```
f <- "./data/fuero-comun-estados.csv"
microbenchmark(
  read.csv(f),
  read_csv(f),
  times = 5) %>%
  print()
```

```
## Unit: seconds
##          expr      min       lq     mean   median       u
##  read.csv(f) 3.229295 3.254810 3.266305 3.279038 3.27918
##  read_csv(f) 1.015490 1.021111 1.171756 1.028151 1.15996
```

plyr

- separar-aplicar-combinar 'SAC'
- Comparar con apply, lapply, (lo-que-sea)ply de R
- Paralelización trivial (.parallel = TRUE)
- `https://github.com/hadley/plyr`
- Funciones importantes: **ply
- La primera letra se refiere al formato de entrada: l (list), d (dataframe), a (array)
- La segunda letra se refiere al formato de salida: l, d, a, _ (nada)
- e.g., ldply(.data = misdatos, .fun = mifuncionqueregresauncuadrodedatos)

plyr

```
iris %>% tbl_df()
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa

plyr

```
lin_mod <-
  ddply(.data = iris,
        .variables = c("Species"),
        .fun = summarise,
        mean_sep_len = mean(Sepal.Length))

lin_mod
```

Species	mean_sep_len
setosa	5.006
versicolor	5.936
virginica	6.588

tidyr

- Reestructurar datos
- `https://github.com/hadley/tidyr`
- Principales funciones: gather, spread

tidyr

```
messy <- data.frame(
  name = c("Botas", "Botellas", "Copetes"),
  a = c(67, 80, 64),
  b = c(56, 90, 50)
)

messy
```

name	a	b
Botas	67	56
Botellas	80	90
Copetes	64	50

tidyr

```
messy %>%
  gather(drug, heartrate, a:b)
```

name	drug	heartrate
Botas	a	67
Botellas	a	80
Copetes	a	64
Botas	b	56
Botellas	b	90
Copetes	b	50

dplyr

- SAC para cuadros de datos
- `https://github.com/hadley/dplyr`
- Gran velocidad
- Funciones:

1. select(): se enfoca en un subconjunto de variables
2. filter(): se enfoca en un subconjunto de renglones
3. mutate(): agrega columnas nuevas
4. summarise(): reduce cada grupo a un número más pequeño a través de estadísticos (e.g., media, varianza)
5. arrange(): reordena los renglones

dplyr

```
iris %>%
  group_by(Species) %>%
  summarise(mean_Sepal_Width = mean(Sepal.Width))
```

Species	mean_Sepal_Width
setosa	3.428
versicolor	2.770
virginica	2.974

dplyr

```
iris %>%
  group_by(Species) %>%
  mutate(Difference = Sepal.Width - Petal.Width) %>%
  select(Species, Difference) %>%
  arrange(Difference)
```

Species	Difference
setosa	2.0
setosa	2.7
setosa	2.7
setosa	2.8
setosa	2.8
setosa	2.8
setosa	2.8
setosa	2.9
setosa	2.9
setosa	2.9

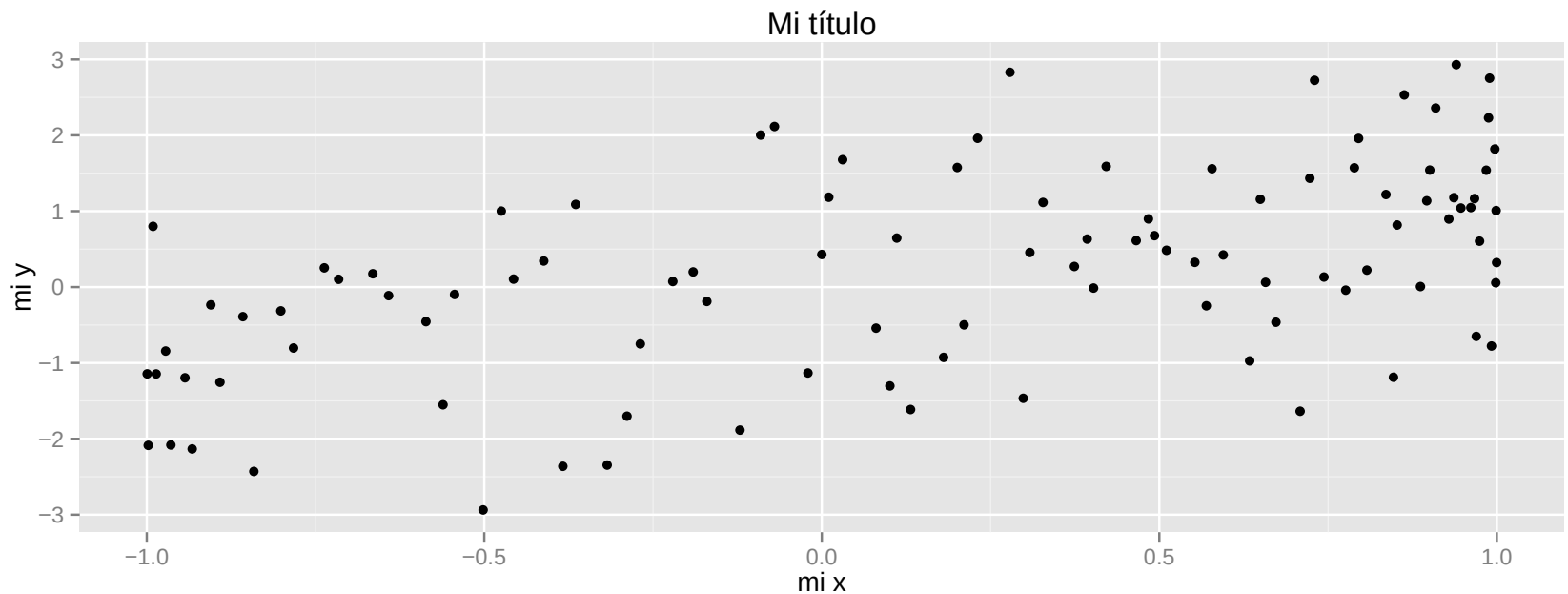
ggplot2

- Gráficas fáciles, elegantes y flexibles
- `https://github.com/hadley/ggplot2`
- Functiones principales: ggplot, qplot y muchas más
- Ver: `http://docs.ggplot2.org/current/`

ggplot2

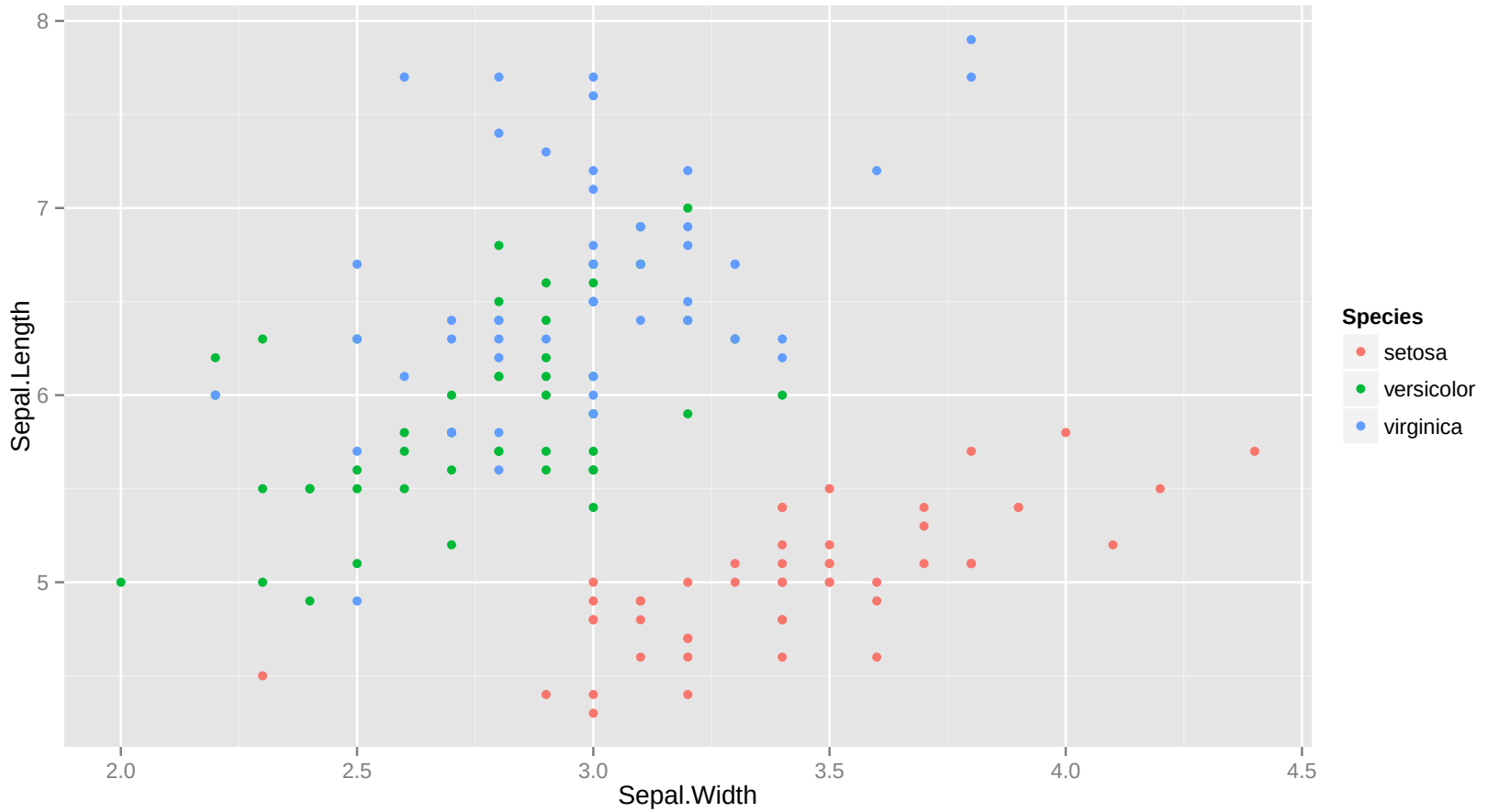
```
x  <- seq(0, 10, len = 100) %>% sin ()
y <- x  + rnorm(length(x))

qplot(x, y,
      main = "Mi título",
      sub = "Mi subtitulo",
      xlab = "mi x",
      ylab = "mi y")
```



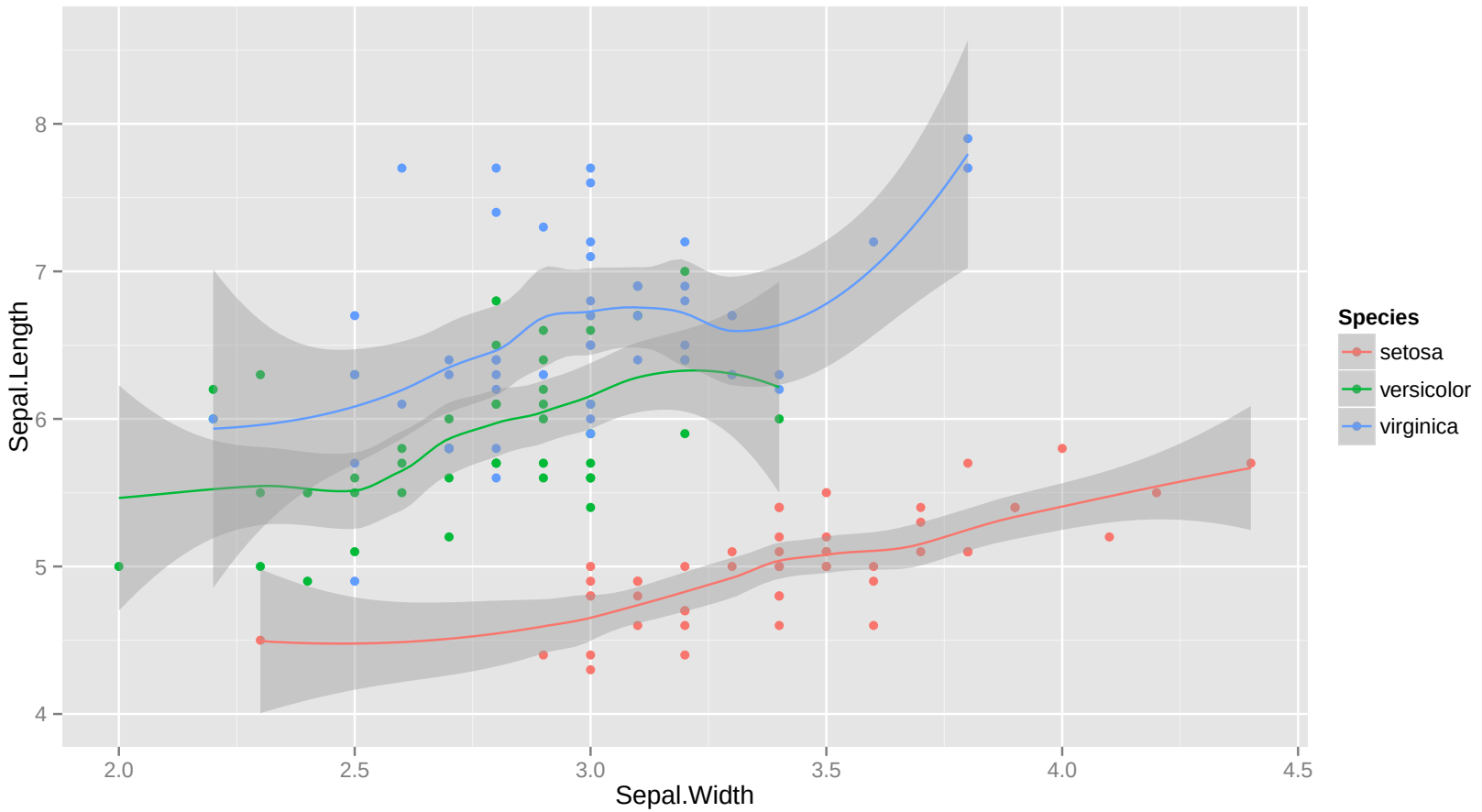
ggplot2

```
ggplot(iris, aes(x = Sepal.Width, y= Sepal.Length, colour =
  geom_point()
```



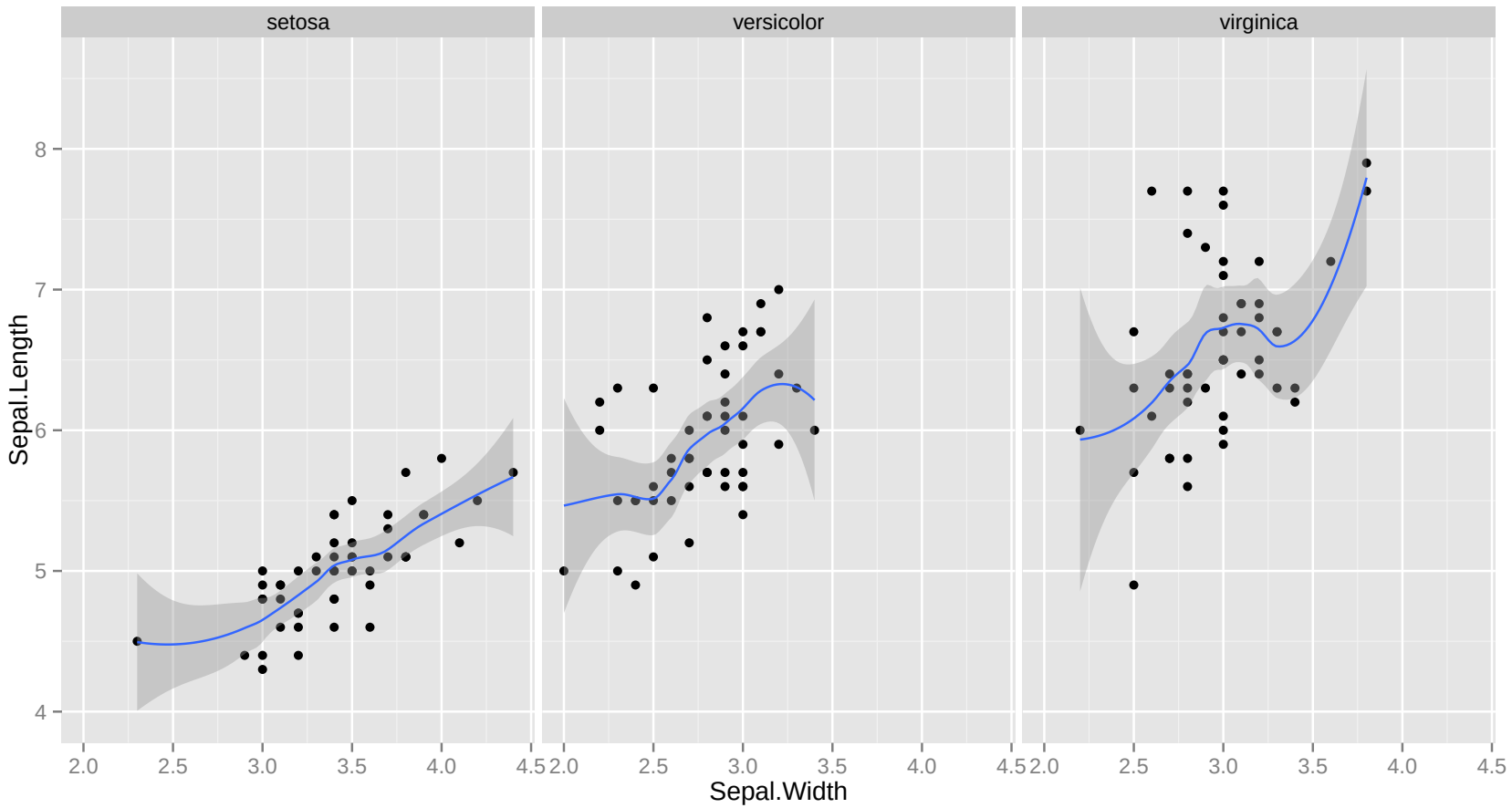
ggplot2

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, colour
  geom_point() +
  stat_smooth()
```



ggplot2

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) +
  geom_point() +
  stat_smooth() +
  facet_wrap(~Species)
```



ggvis

- Gráficas interactivas
- Su propósito es combinar las principals ideas de ggplot2, shiny y dplyr
- `https://github.com/rstudio/ggvis`
- `http://ggvis.rstudio.com/`

Rcpp

- Llamar C++ desde R
- Fácil accceso a GSL, Armadillo, etc
- `http://www.rcpp.org/`

lubridate

- Manejo de fechas
- `https://github.com/hadley/lubridate`
- Fácil de usar con notación muy intuitiva
- Horarios

lubridate

```
ymd("20110604")
```

```
## [1] "2011-06-04 UTC"
```

```
mdy("06-04-2011")
```

```
## [1] "2011-06-04 UTC"
```

```
dmy("04/06/2011")
```

```
## [1] "2011-06-04 UTC"
```

lubridate

```
now("Pacific/Auckland")
```

```
## [1] "2015-03-21 06:23:48 NZDT"
```

```
today()
```

```
## [1] "2015-03-20"
```

lubridate

```
arrive <- ymd_hms("2011-06-04 12:00:00", tz = "Pacific/Auck
leave <- ymd_hms("2011-08-10 14:00:00", tz = "Pacific/Auckl

second(arrive)
```

```
## [1] 0
```

```
wday(arrive)
```

```
## [1] 7
```

```
wday(arrive, label = TRUE)
```

```
## [1] Sat
## Levels: Sun < Mon < Tues < Wed < Thurs < Fri < Sat
```

lubridate

```
arrive <- ymd_hms("2011-06-04 12:00:00", tz = "Pacific/Auck
leave <- ymd_hms("2011-08-10 14:00:00", tz = "Pacific/Auckl

leave - arrive
```

```
## Time difference of 67.08333 days
```

```
leave + days(2)
```

```
## [1] "2011-08-12 14:00:00 NZST"
```

Referencias

- Las referencias y algunos de los ejemplos dados en esta plática fueron obtenidos de los hiperenlaces mencionados