PEC1 Análisis de datos ómicos

Luis Morís Fernández

2020-04-19

Información sobre el dataset

Vamos a utilizar el dataset GSE7614. Este dataset corresponde al estudio *Phenotypic and transcriptional response to selection for alcohol sensitivity in Drosophila melanogaster*.

Información del dataset	
Organismo	Drosophila Melanogaster
Número de muestras	24 (15 individuos por muestra)
Grupos	Macho Control (x2 Rep. Biológicas x2 Líneas) Hembra Control (x2 Rep. Biológicas x2 Líneas) Macho Sensible (x2 Rep. Biológicas x2 Líneas) Hembra Sensible (x2 Rep. Biológicas x2 Líneas) Macho Resistente (x2 Rep. Biológicas x2 Líneas) Hembra Resistente (x2 Rep. Biológicas x2 Líneas)
Número de características	18952
Plataforma	[Drosophila_2] Affymetrix Drosophila Genome 2.0 Array GPL1322
Tipo de muestra	*in situ oligonucleotide*

```
## Platform design info loaded.
```

```
## Reading in : ./data//GSM183815.CEL
## Reading in : ./data//GSM183816.CEL
## Reading in : ./data//GSM183817.CEL
## Reading in : ./data//GSM183856.CEL
## Reading in : ./data//GSM183857.CEL
## Reading in : ./data//GSM183858.CEL
## Reading in : ./data//GSM183875.CEL
## Reading in : ./data//GSM183886.CEL
## Reading in : ./data//GSM183887.CEL
## Reading in : ./data//GSM183888.CEL
## Reading in : ./data//GSM183936.CEL
## Reading in : ./data//GSM183989.CEL
## Reading in : ./data//GSM183990.CEL
## Reading in : ./data//GSM183991.CEL
## Reading in : ./data//GSM183992.CEL
## Reading in : ./data//GSM183993.CEL
## Reading in : ./data//GSM183994.CEL
```

```
## Reading in : ./data//GSM184118.CEL
## Reading in : ./data//GSM184119.CEL
## Reading in : ./data//GSM184120.CEL
## Reading in : ./data//GSM184121.CEL
## Reading in : ./data//GSM184122.CEL
## Reading in : ./data//GSM184123.CEL
## Reading in : ./data//GSM184124.CEL

## Warning in read.celfiles(cel_files, phenoData = my_targets): 'channel'
## automatically added to varMetadata in phenoData.
```

Cargado de datos crudos

En este caso hemos descargado todos los archivos .CEL del dataset GSE7614. [1] También hemos consultado a que grupo pertenece cada una de las 24 muestras que contiene el dataset. Hemos preparado también un archivo que contiene toda la información referente a los grupos y las muestras y lo hemos cargado en memoria.

[1] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE7614

Table 2: Muestra del archivo que contiene *phenoData*

sample	file	selection	line	sex	short_name	group
GSM183815	GSM183815.CEL	control	1	male	control_1_male_1	control_male
GSM183816	GSM183816.CEL	control	1	male	control_1_male_2	control_male
GSM183817	GSM183817.CEL	control	1	female	control_1_female_1	control_female
GSM183856	GSM183856.CEL	control	1	female	control_1_female_2	control_female
GSM183857	GSM183857.CEL	sensitive	2	male	sensitive_2_male_1	sensitive_male
GSM183858	GSM183858.CEL	sensitive	2	male	sensitive_2_male_2	sensitive_male

Luego hemos generado un ExpressionFeatureSet juntando los archivos .CEL y annotatedDataFrame que contiene el *phenoData*

```
## ExpressionFeatureSet (storageMode: lockedEnvironment)
## assayData: 535824 features, 24 samples
##   element names: exprs
## protocolData
##   rowNames: control_1_male_1 control_1_male_2 ... resistant_6_female_2
##     (24 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: control_1_male_1 control_1_male_2 ... resistant_6_female_2
##     (24 total)
##   varLabels: file selection ... group (6 total)
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
```

```
## Annotation: pd.drosophila.2

## Loading required package: pd.drosophila.2

## Loading required package: RSQLite

## Loading required package: DBI
```

Vamos ahora a hacer un control de calidad de estos datos crudos.

Control de calidad de los datos crudos

```
# Lo hacemos con el Array Quality Metrics, queda comentados para no ejecutarlos continuamente
# Luego sacaremos los gráficos de ahí
# arrayQualityMetrics::arrayQualityMetrics(raw_data, intgroup = c('sex', 'selection', 'line'), outdir =
# arrayQualityMetrics::arrayQualityMetrics(raw_data, intgroup = c('sex', 'selection', 'line'), outdir =
# exprs(raw_data)
```

Para realizar un control de la calidad de los datos crudos vamos utilizar la libreria Array Quality Metrics y affysimple.

Lo haremos de dos maneras diferentes, primero sin transformar los datos al logaritmo de la intensidad y luego haciéndolo.

Sin transformar

- En el gráfico A podemos observar un resumen de todas las muestras y el resultado obtenido del control de outliers. Vemos que según el algoritmo las muestras 1 y 17 podrían presentar algún problema por ello las tendremos en especial consideración cuando observemos los gráficos. En este caso no disponemos de la posibilidad de hacerlo, pero podríamos comprobar las imágenes de los microarrays para ver si ha ocurrido algún problema durante su procesado.
- En el gráfico B observamos una agrupación por distancias de los microarrays, vemos que hay una separación clara según el sexo de la muestra, a excepción de las muestras 1 y 17 que muestran una distancia un poco atípica se encuentran más distantes del resto de muestras de machos.
- En el gráfico C observamos los boxplots de las intensidades de cada uno de los arrays. También aparece marcado el array quizá más diperso que el resto, también aparece marcado el 10, sin embargo no parece que sean muestras particularmente aberrantes.
- En el gráfico D podemos observar un análisis por componentes principales, veremos debajo uno como el que se ha realizado durante las prácticas de la asignatura, en este caso vemos una separación

clara entre los grupos macho y hembra de las muestras lo cual es esperable y deseable.

- En el gráfico E, vemos que las muestra siguen un mismo patrón, aunque la escala hace un poco complicada esta observación, probaremos luego con una función de affysimple.
- En el gráfico F, el valor esperable es que la mediana sea constante, sin embargo aparece una pequeña joroba en el lado derecho, pero seguramente esté relacionado con una saturación de la intensidades como indica el propio paquete.
- En el gráfico G, podemos observar la comparación de cada uno de los arrays respecto a un pseudo-array que tiene el valor mediano a través del resto de arrays. En este caso lo esperable es que los valores se concentren a lo largo del eje M=0. Sin embargo al no estar transformados los valores aparece agrupados en el margen izquierdo haciendo difícil la visualización.

  Vamos ahora a ver un PCA más completo

Log-Transform

Al estar transformados con el logaritmo la forma de las gráficas cambia ya que hemos *estirado* la distribución. El significado de los gráficos es el mismo que en el apartado anterior.

El array 17 sigue apareciendo como un outlier, sin embargo en ninguno de los plots aparece particularmente dramático por tanto lo conservaremos.

affyPLM

Vamos ahora a utilizar la librería *affyPLM* vamos a ajusjar un modelo a nivel de sonda y vamos a comprobar las gráficas *RLE* y *NUSE*.

En el caso del *RLE* vamos a encontrar una medida estandarizada de la expresión, que ha de ser similar a través de arrays, y en el caso de *NUSE* observaremos los residuos normalizados del modelo cuya distribución ha de ser similar a través de arrays.

No parece que ninguno de los arrays se desvíe particularmente en ninguna de las dos gráficas, si que vemos que hay una diferencia clara entre sexos. Podríamos preguntarnos en este caso si se ha cometido algún error y los machos perteneces a un batch y las hembras a otros, es decir hemos generado algún tipo de confound, aunque es probable que se deba a diferencias reales.

Conclusiones de calidad

- Conservamos todos los arrays

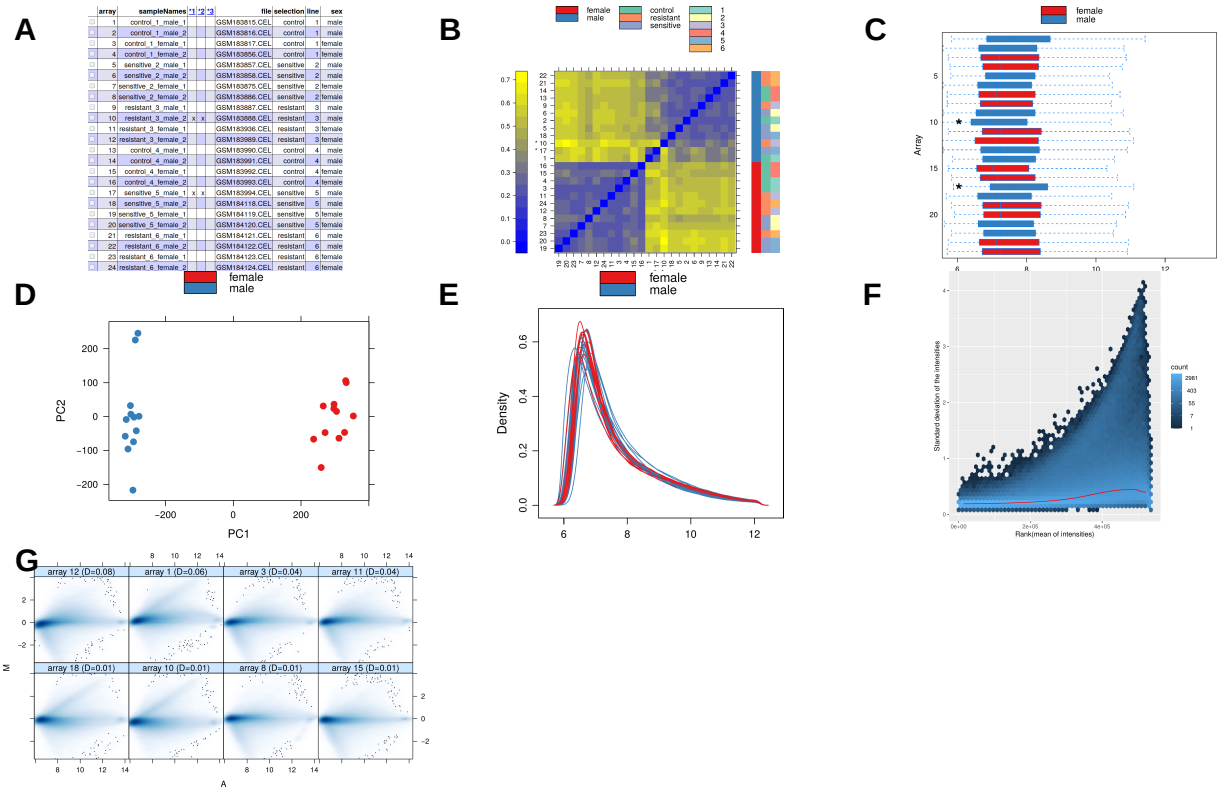


Figure 1: Array Quality Metrics Output

- Podría ser útil revisar el 17
- Podría ser útil consultar si ha habido alguna diferencia sistemática entre los arrays de machos y hembras, batch, fecha, operario,…

Normalización

Procedemos ahora a normalizar los datos con el método *RMA*.

Durante este paso el algoritmo:

- Eliminará el ruido de fondo
- Normalizará la señal entre arrays, nunca entre condiciones experimentales.
- Colapsará todos los valores de cada grupo de sondas en un único valor por gen.

```
if(!file.exists('rma_normalized.Rda')){
  eset_rma = oligo::rma(raw_data)
  save(eset_rma, file="rma_normalized.Rda")
}else{
  load(file="rma_normalized.Rda")
}
```

```
# arrayQualityMetrics::arrayQualityMetrics(eset_rma, intgroup = c('sex', 'selection', 'line'), outdir =
```

Una vez normalizado vamos a volver a realizar un análisis de calidad de los datos

Array quality metrics output después de la normalización

Los gráficos son los mismos que en apartados anteriores, ahora vemos una separación mucho más clara en el array de distancias. El array número 17 ahora es más similar al resto.

Solo existen dos detalles destacables, uno es la diferencia entre patrones de intensidad en la figura E entre las muestras machos y hembras, y la joroba que aparece en la figura F, cuando la mediana debería de ser plana, quizá sea debida a la diferencia entre sexos.

Conclusiones normalización

Los datos parecen correctos, por tanto no es necesaria ninguna acción.

Filtrado

No se va a realizar ningún filtrado no específico.

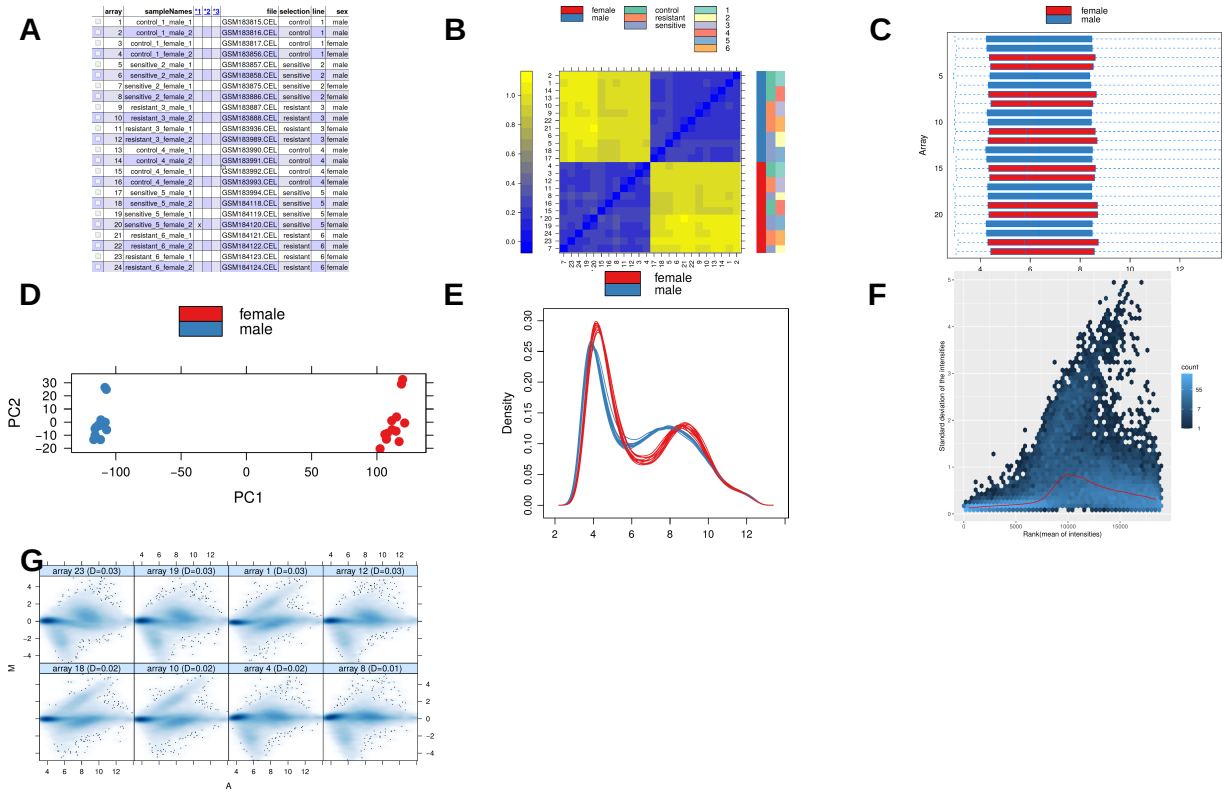


Figure 2: Array Quality Metrics Output

Comparaciones

Estamos ante um diseño de 3x2, en este caso vamos a simular que estamos interesados en las comparaciones:

- Main Effect Macho vs. Hembra
- Single Effect Resistente vs Sensible
- Interacción Sexo vs (Resistente vs Sensible)

```
design_matrix = model.matrix(~0+group, data=pData(eset_rma))
colnames(design_matrix)=c('control_female','control_male','resistant_female','resistant_male','sensitiv

contrast_matrix = makeContrasts(res_vs_sens = resistant_female + resistant_male - sensitive_female - se
                                male_vs_female = resistant_female + control_female + sensitive_female -
                                int_sex_resvssens = (resistant_female - resistant_male) - (sensitive_fe
                                levels = design_matrix)

m = lmFit(eset_rma, design_matrix)
cm = eBayes(contrasts.fit(m, contrast_matrix))
top_sex = topTable(cm, coef='male_vs_female', adjust='fdr')
top_res_vs_ses = topTable(cm, coef='res_vs_sens', adjust='fdr')
top_int = topTable(cm, coef='int_sex_resvssens', adjust='fdr')
```

Anotación

```
library(drosophila2.db)
```

```
## Loading required package: AnnotationDbi
```

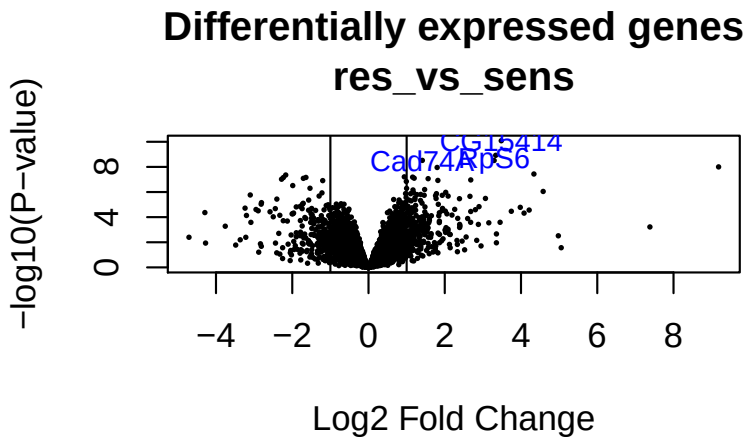
```
## Loading required package: org.Dm.eg.db
```

```
##
```

```
##
```

```
annotatedTopTable <- function(topTab, anotPackage)
{
  topTab <- cbind(PROBEID=rownames(topTab), topTab)
  myProbes <- rownames(topTab)
  thePackage <- eval(parse(text = anotPackage))
  geneAnots <- select(thePackage, myProbes, c("SYMBOL", "ENTREZID", "GENENAME"))
  annotatedTopTab<- merge(x=geneAnots, y=topTab, by.x="PROBEID", by.y="PROBEID")
return(annotatedTopTab)
}

top_sex_annotated = annotatedTopTable(top_sex, anotPackage = 'drosophila2.db')
```

```
## 'select()' returned 1:1 mapping between keys and columns

top_int = annotatedTopTable(top_int, anotPackage = 'drosophila2.db')

## 'select()' returned 1:1 mapping between keys and columns

top_res_vs_ses = annotatedTopTable(top_res_vs_ses, anotPackage = 'drosophila2.db')

## 'select()' returned 1:1 mapping between keys and columns

geneSymbols <- select(drosophila2.db, rownames(cm), c("SYMBOL"))

## 'select()' returned 1:many mapping between keys and columns

SYMBOLS<- geneSymbols$SYMBOL
volcanoplot(cm, coef=1, highlight=4, names=SYMBOLS,
            main=paste("Differentially expressed genes", colnames(contrast_matrix)[1], sep="\n"))
  abline(v=c(-1,1))
```



Introduction

The Tufte handout style is a style that Edward Tufte uses in his books and handouts. Tufte's style is known for its extensive use of sidenotes, tight integration of graphics with text, and well-set typography. This style has been implemented in LaTeX and HTML/CSS[2], respectively. We have ported both implementations into the **tufte** package. If you want LaTeX/PDF output, you may use the `tufte_handout` format for handouts, and `tufte_book` for books. For HTML output, use `tufte_html`. These formats can be either specified in the YAML metadata at the beginning of an R Markdown document (see an example below), or passed to the `rmarkdown::render()` function. See Allaire et al. [2020] for more information about **rmarkdown**.

[2] See Github repositories tufte-latex and tufte-css

```
---
title: "An Example Using the Tufte Style"
author: "John Smith"
output:
  tufte::tufte_handout: default
  tufte::tufte_html: default
---
```

There are two goals of this package:

1. To produce both PDF and HTML output with similar styles from the same R Markdown document;
2. To provide simple syntax to write elements of the Tufte style such as side notes and margin figures, e.g. when you want a margin figure, all you need to do is the chunk option `fig.margin = TRUE`, and we will take care of the details for you, so you never need to think about `\begin{marginfigure} \end{marginfigure}` or `<span class="marginfigure"> </span>`; the LaTeX and HTML code under the hood may be complicated, but you never need to learn or write such code.

If you have any feature requests or find bugs in **tufte**, please do not hesitate to file them to `https://github.com/rstudio/tufte/issues`. For general questions, you may ask them on StackOverflow: `http://stackoverflow.com/tags/rmarkdown`.

Headings

This style provides first and second-level headings (that is, `#` and `##`), demonstrated in the next section. You may get unexpected output if you try to use `###` and smaller headings.

In his later books[3], Tufte starts each section with a bit of vertical space, a non-indented paragraph, and sets the first few words of the sentence in small caps. To accomplish this using this style, call the `newthought()` function in **tufte** in an *inline R expression* `` `r ` `` as demonstrated at the beginning of this paragraph.[4]

[3] Beautiful Evidence

[4] Note you should not assume **tufte** has been attached to your R session. You should either `library(tufte)` in your R Markdown document before you call `newthought()`, or use `tufte::newthought()`.

Figures

Margin Figures

Images and graphics play an integral role in Tufte's work. To place figures in the margin you can use the **knitr** chunk option `fig.margin = TRUE`. For example:

```r
library(ggplot2)
mtcars2 <- mtcars
mtcars2$am <- factor(
  mtcars$am, labels = c('automatic', 'manual')
)
ggplot(mtcars2, aes(hp, mpg, color = am)) +
  geom_point() + geom_smooth() +
  theme(legend.position = 'bottom')
```

Note the use of the `fig.cap` chunk option to provide a figure caption. You can adjust the proportions of figures using the `fig.width` and `fig.height` chunk options. These are specified in inches, and will be automatically scaled down to fit within the handout margin.

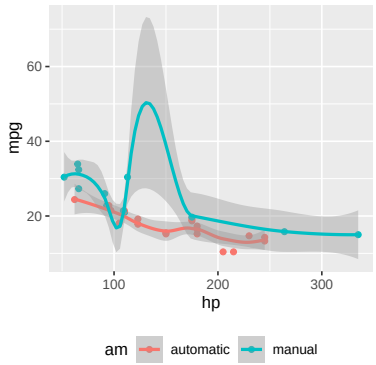


Figure 3: MPG vs horsepower, colored by transmission.

Arbitrary Margin Content

In fact, you can include anything in the margin using the **knitr** engine named `marginfigure`. Unlike R code chunks ```` ```{r} ````, you write a chunk starting with ```` ```{marginfigure} ```` instead, then put the content in the chunk. See an example on the right about the first fundamental theorem of calculus.

For the sake of portability between LaTeX and HTML, you should keep the margin content as simple as possible (syntax-wise) in the `marginefigure` blocks. You may use simple Markdown syntax like `**bold**` and `_italic_` text, but please refrain from using footnotes, citations, or block-level elements (e.g. blockquotes and lists) there.

Note: if you set `echo = FALSE` in your global chunk options, you will have to add `echo = TRUE` to the chunk to display a margin figure, for example ```` ```{marginfigure, echo = TRUE} ````.

We know from *the first fundamental theorem of calculus* that for x in $[a, b]$:

$$\frac{d}{dx}\left(\int_a^x f(u)\,du\right) = f(x).$$

Full Width Figures

You can arrange for figures to span across the entire page by using the chunk option `fig.fullwidth = TRUE`.

```r
ggplot(diamonds, aes(carat, price)) + geom_smooth() +
  facet_grid(~ cut)
```

Other chunk options related to figures can still be used, such as `fig.width`, `fig.cap`, `out.width`, and so on. For full width figures, usually `fig.width` is large and `fig.height` is small. In the above example, the plot size is 10×2.

Main Column Figures

Besides margin and full width figures, you can of course also include figures constrained to the main column. This is the default type of

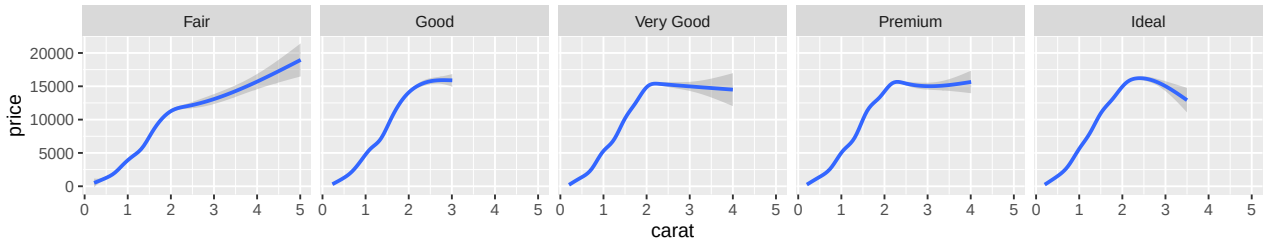


Figure 4: A full width figure.

figures in the LaTeX/HTML output.

```
ggplot(diamonds, aes(cut, price)) + geom_boxplot()
```

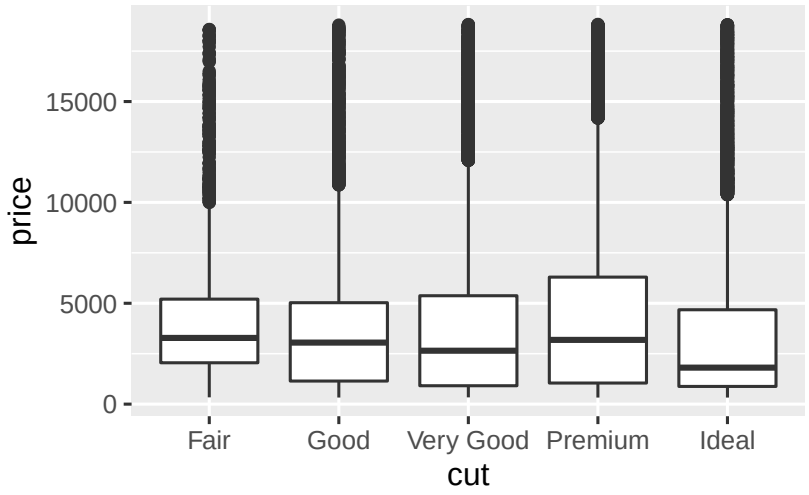


Figure 5: A figure in the main column.

Sidenotes

One of the most prominent and distinctive features of this style is the extensive use of sidenotes. There is a wide margin to provide ample room for sidenotes and small figures. Any use of a footnote will automatically be converted to a sidenote. [5]

If you'd like to place ancillary information in the margin without the sidenote mark (the superscript number), you can use the `margin_note()` function from **tufte** in an inline R expression. This function does not process the text with Pandoc, so Markdown syntax will not work here. If you need to write anything in Markdown syntax, please use the `marginfigure` block described previously.

[5] This is a sidenote that was entered using a footnote.

This is a margin note. Notice that there is no number preceding the note.

References

References can be displayed as margin notes for HTML output. For example, we can cite R here [R Core Team, 2020]. To enable this feature, you must set `link-citations: yes` in the YAML metadata, and the version of `pandoc-citeproc` should be at least 0.7.2. You can always install your own version of Pandoc from `http://pandoc.org/installing.html` if the version is not sufficient. To check the version of `pandoc-citeproc` in your system, you may run this in R:

```
system2('pandoc-citeproc', '--version')
```

If your version of `pandoc-citeproc` is too low, or you did not set `link-citations: yes` in YAML, references in the HTML output will be placed at the end of the output document.

Tables

You can use the `kable()` function from the **knitr** package to format tables that integrate well with the rest of the Tufte handout style. The table captions are placed in the margin like figures in the HTML output.

```
knitr::kable(
  mtcars[1:6, 1:6], caption = 'A subset of mtcars.'
)
```

Table 3: A subset of mtcars.

	mpg	cyl	disp	hp	drat	wt
Mazda RX4	21.0	6	160	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875
Datsun 710	22.8	4	108	93	3.85	2.320
Hornet 4 Drive	21.4	6	258	110	3.08	3.215
Hornet Sportabout	18.7	8	360	175	3.15	3.440
Valiant	18.1	6	225	105	2.76	3.460

Block Quotes

We know from the Markdown syntax that paragraphs that start with `>` are converted to block quotes. If you want to add a right-aligned footer for the quote, you may use the function `quote_footer()` from **tufte** in an inline R expression. Here is an example:

> "If it weren't for my lawyer, I'd still be in prison. It went a lot faster with two people digging."
>
> — Joe Martin

Without using `quote_footer()`, it looks like this (the second line is just a normal paragraph):

> "Great people talk about ideas, average people talk about things, and small people talk about wine."
>
> — Fran Lebowitz

Responsiveness

The HTML page is responsive in the sense that when the page width is smaller than 760px, sidenotes and margin notes will be hidden by default. For sidenotes, you can click their numbers (the superscripts) to toggle their visibility. For margin notes, you may click the circled plus signs to toggle visibility.

More Examples

The rest of this document consists of a few test cases to make sure everything still works well in slightly more complicated scenarios. First we generate two plots in one figure environment with the chunk option `fig.show = 'hold'`:

```
p <- ggplot(mtcars2, aes(hp, mpg, color = am)) +
  geom_point()
p
p + geom_smooth()
```

Then two plots in separate figure environments (the code is identical to the previous code chunk, but the chunk option is the default `fig.show = 'asis'` now):

```
p <- ggplot(mtcars2, aes(hp, mpg, color = am)) +
  geom_point()
p
```

```
p + geom_smooth()
```

You may have noticed that the two figures have different captions, and that is because we used a character vector of length 2 for the chunk option `fig.cap` (something like `fig.cap = c('first plot', 'second plot')`).

Next we show multiple plots in margin figures. Similarly, two plots in the same figure environment in the margin:

```
p
p + geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula 'y ~ x'
```

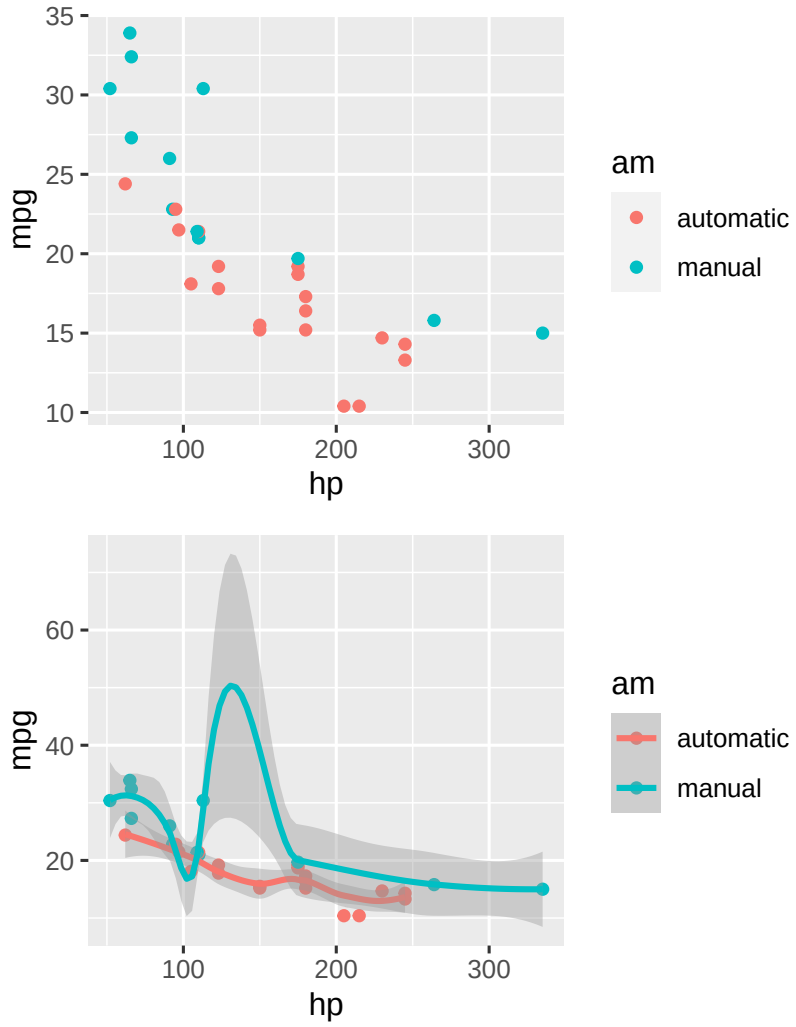


Figure 6: Two plots in one figure environment.

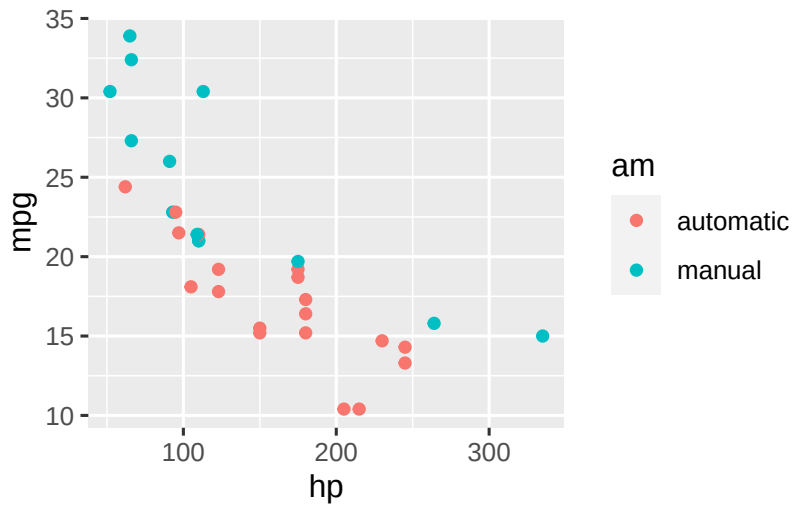


Figure 7: Two plots in separate figure environments (the first plot).

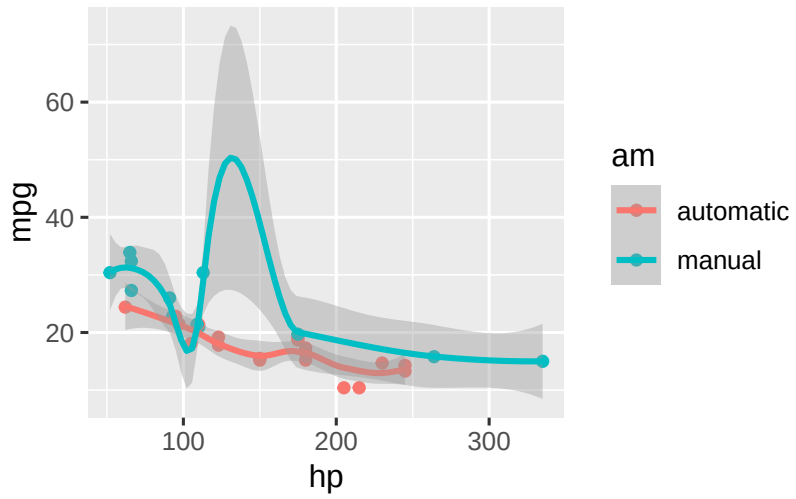


Figure 8: Two plots in separate figure environments (the second plot).

Then two plots from the same code chunk placed in different figure environments:

```
knitr::kable(head(iris, 15))
```

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
5.8	4.0	1.2	0.2	setosa

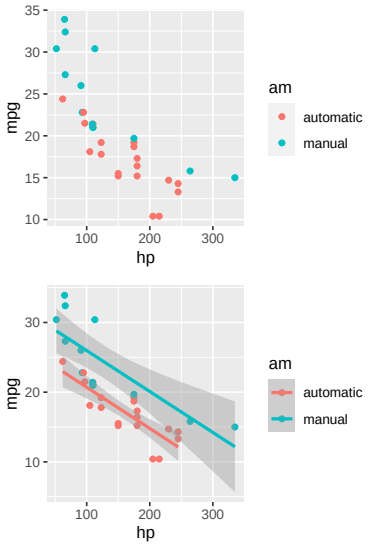


Figure 9: Two plots in one figure environment in the margin.

```
p
```

```
knitr::kable(head(iris, 12))
```

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

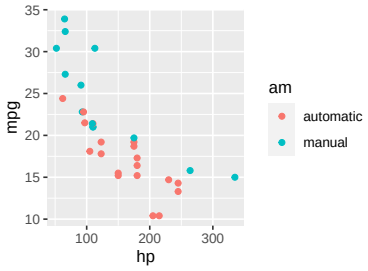


Figure 10: Two plots in separate figure environments in the margin (the first plot).

```
p + geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
knitr::kable(head(iris, 5))
```

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

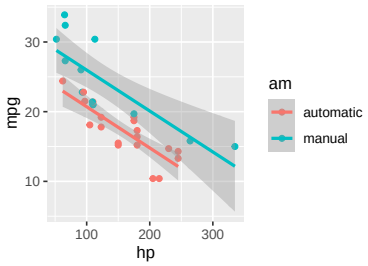
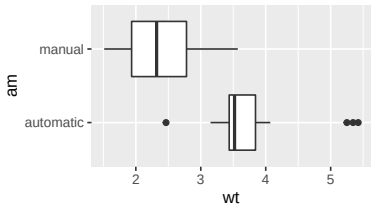


Figure 11: Two plots in separate figure environments in the margin (the second plot).

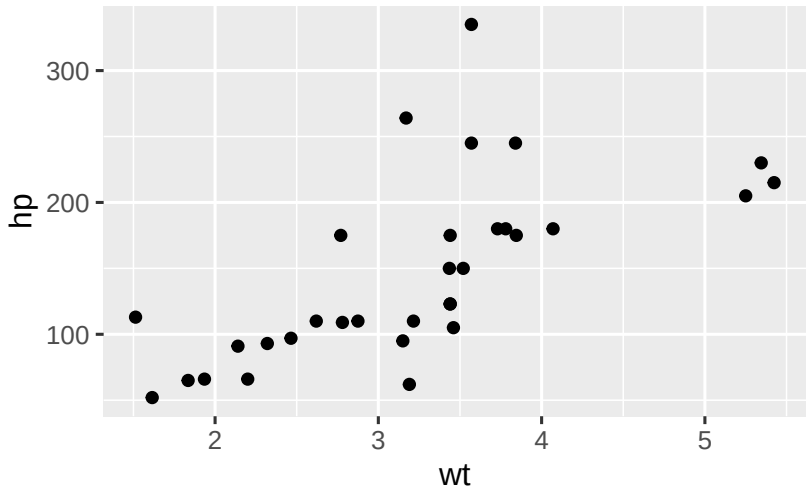
We blended some tables in the above code chunk only as *placeholders* to make sure there is enough vertical space among the margin figures, otherwise they will be stacked tightly together. For a practical document, you should not insert too many margin figures consecutively and make the margin crowded.

You do not have to assign captions to figures. We show three figures with no captions below in the margin, in the main column, and in full width, respectively.

```
# a boxplot of weight vs transmission; this figure
# will be placed in the margin
ggplot(mtcars2, aes(am, wt)) + geom_boxplot() +
  coord_flip()
```

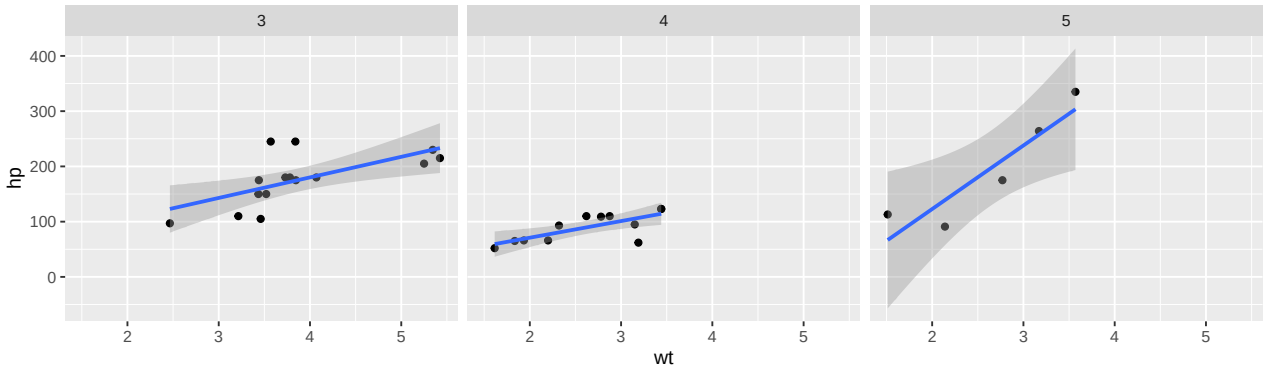



```
# a figure in the main column
p <- ggplot(mtcars, aes(wt, hp)) + geom_point()
p
```




```
# a fullwidth figure
p + geom_smooth(method = 'lm') + facet_grid(~ gear)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Some Notes on Tufte CSS

There are a few other things in Tufte CSS that we have not mentioned so far. If you prefer sans-serif fonts, use the function `sans_serif()` in **tufte**. For epigraphs, you may use a pair of underscores to make the paragraph italic in a block quote, e.g.

> *I can win an argument on any topic, against any opponent. People know this, and steer clear of me at parties. Often, as a sign of their great respect, they don't even invite me.*
>
> — Dave Barry

We hope you will enjoy the simplicity of R Markdown and this R package, and we sincerely thank the authors of the Tufte-CSS and Tufte-LaTeX projects for developing the beautiful CSS and LaTeX classes. Our **tufte** package would not have been possible without their heavy lifting.

You can turn on/off some features of the Tufte style in HTML output. The default features enabled are:

```
output:
  tufte::tufte_html:
    tufte_features: ["fonts", "background", "italics"]
```

If you do not want the page background to be lightyellow, you can remove `background` from `tufte_features`. You can also customize the style of the HTML page via a CSS file. For example, if you do not want the subtitle to be italic, you can define

```
h3.subtitle em {
  font-style: normal;
}
```

in, say, a CSS file `my_style.css` (under the same directory of your Rmd document), and apply it to your HTML output via the `css` option, e.g.,

```
output:
  tufte::tufte_html:
    tufte_features: ["fonts", "background"]
    css: "my_style.css"
```

There is also a variant of the Tufte style in HTML/CSS named "Envisoned CSS". This style can be used by specifying the argument `tufte_variant = 'envisioned'` in `tufte_html()`[6], e.g.

```
output:
  tufte::tufte_html:
    tufte_variant: "envisioned"
```

To see the R Markdown source of this example document, you may follow this link to Github, use the wizard in RStudio IDE (`File -> New File -> R Markdown -> From Template`), or open the Rmd file in the package:

```
file.edit(
  tufte:::template_resources(
    'tufte_html', '..', 'skeleton', 'skeleton.Rmd'
  )
)
```

[6] The actual Envisioned CSS was not used in the **tufte** package. We only changed the fonts, background color, and text color based on the default Tufte style.

This document is also available in Chinese, and its `envisioned` style can be found here.

References

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. *rmarkdown: Dynamic Documents for R*, 2020. URL `https://CRAN.R-project.org/package=rmarkdown`. R package version 2.1.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL `https://www.R-project.org/`.