{\rtf1\ansi\ansicpg1252\cocoartf1504\cocoasubrtf820  
{\fonttbl\f0\fswiss\fcharset0 Helvetica;}  
{\colortbl;\red255\green255\blue255;}  
{\\*\expandedcolortbl;;}  
\paperw11900\paperh16840\margl1440\margr1440\vieww10800\viewh8400\viewkind0  
\pard\tx566\tx1133\tx1700\tx2267\tx2834\tx3401\tx3968\tx4535\tx5102\tx5669\tx6236\tx6803\pardirnatural\partightenfactor0  
  
\f0\fs24 \cf0 ---\  
title: "Introduzione ad R-Markdown"\  
author: "Tullia Padellini"\  
date: "21 Aprile 2017"\  
output: html\_document\  
---\  
\  
## R Markdown\  
Rmarkdown \'e8 uno strumento che permette di costruire documenti in cui combinare testo formattato (quindi non solo commenti ma anche formule, titoli etc) e istruzioni codice (`R` e non solo) con i corrispettivi output.\  
\  
Un file R-markdown \'e8 composto da tre tipi di oggetti: \  
\  
1. Header in formato `YAML` delimitato da `---`\  
3. Testo in formato `markdown` \  
2. Blocchi ("chunks") di codice `R`, delimitati da ` ``` `\  
\  
\  
## Testo \  
\  
#### Formattazione base:\  
\  
Il `markdown` nasce come formato per scrivere documenti `html` in modo "semplificato". \'c8 una sintassi estremamente semplice per creare un documento di testo (talmente semplice che \'e8 quella utilizzata di base addirittura in whatsapp!).\  
\  
Iniziamo dalla formattazione di base: \  
\  
- Per ottenere grassetto e corsivo \'e8 sufficiente racchiudere il testo in una coppia di `\*` o `\_`. In particolare utilizzando un carattere singolo, ossia `\*testo\*` o `\_testo\_` rendiamo \*testo\* corsivo, mentre utilizzando due caratteri ossia `\*\*testo\*\*` o `\_\_testo\_\_` rendiamo \*\*testo\*\* grassetto. Naturalmente con `\*\*\*testo\*\*\*` rendiamo \*\*\*testo\*\*\* grassetto e corsivo. \  
\  
- Per avere del testo ~~barrato~~ possiamo usare il carattere tilda `~` `~~barrato~~`.\  
\  
- Per avere testo^apice^ e testo~pedice~ possiamo invece usare `testo^apice^` e `testo~pedice~`.\  
\  
#### Elenchi & Quotes\  
\  
Per creare una lista in `markdown` \'e8 sufficiente utilizzare i caratteri `\*` o `-` all'inizio di ogni elemento, ad esempio:\  
\  
`\  
\* primo elemento\  
`\  
\  
`\  
\* secondo elemento\  
`\  
\  
`\  
\* terzo elemento\  
`\  
\  
produce la seguente lista \  
\  
\* primo elemento\  
\* secondo elemento\  
\* terzo elemento\  
\  
\  
Possiamo utilizzare il carattere `+` per aggiungere un secondo livello di lista, ad esempio\  
\  
\  
`\  
\* primo elemento\  
`\  
\  
`\  
+ primo sotto-elemento\  
`\  
\  
`\  
+ secondo sottoelemento\  
`\  
\  
`\  
\* secondo elemento\  
`\  
\  
`\  
\* terzo elemento\  
`\  
\  
produce la seguente lista:\  
\  
- primo elemento\  
+ primo sotto-elemento\  
+ secondo sotto-elemento\  
- secondo elemento\  
- terzo elemento\  
\  
\  
Per creare una elenco numerato, stesso procedimento ma utilizzando 1. (o un qualunque altro numero messo a caso, markdown non segue le vostre indicazioni ma numera da solo gli elementi) per specificare i diversi elementi, ad esempio il comando:\  
\  
`\  
1. primo elemento\  
`\  
\  
` + primo sottoelemento\  
`\  
\  
`\  
3. secondo elemento\  
`\  
\  
`\  
1. terzo elemento\  
`\  
\  
produce l'elenco\  
\  
\  
1. primo elemento\  
+ primo sottoelemento\  
3. secondo elemento\  
1. terzo elemento\  
\  
\  
\  
Infine per ottenere del testo in formato\  
\  
> quote\  
\  
\'e8 sufficiente scrivere `>` quote. \  
\  
\  
#### Link & Immagini\  
\  
Per inserire un link ci sono due metodi: \  
\  
- specificare solo il percorso `<<http://rmarkdown.rstudio.com>>`, <<http://rmarkdown.rstudio.com>>\  
- creare un [link](<http://rmarkdown.rstudio.com>) con `[link](<http://rmarkdown.rstudio.com>)`\  
\  
\  
Per inserire un'immagine la sintassi \'e8 molto simile: `![image](correlation.png)`. \  
Un esempio?\  
![image](correlation.png)\  
\  
#### Sezioni\  
\  
I titoli delle sezioni in markdown sono detti "header", in particolare esistono 6 diversi livelli di header, tutti definiti dal carattere `#`\  
\  
- `# Header di primo livello`\  
- `## Header di secondo livello`\  
- ...\  
- `###### Header di sesto livello`\  
\  
che nel documento finale si presentano come segue:\  
\  
\  
# Header di primo livello\  
\  
## Header di secondo livello \  
\  
### Header di terzo livello \  
\  
#### Header di quarto livello \  
\  
##### Header di quinto livello \  
\  
###### Header di sesto livello\  
\  
\  
\  
#### Formule\  
R-markdown ci permette di inserire espressioni matematiche nel documento `hmtl` grazie al cosiddetto \*\*ambiente matematico\*\*. \  
\  
L'ambiente matematico \'e8 delimitato dal simbolo `$`. Tecnicamente l'ambiente matematico richiede una nuova sintassi, che non \'e8 n\'e8 quella di `R`, n\'e8 quella del `markdown`, ma si chiama `LaTeX` (per maggiori dettagli [link](<https://www.latex-project.org/>)).\  
Possiamo distinguere ambiente matematico inline `$x+y$`, ossia $x+y$ da ambiente matematico "centrato" `$$x+y$$`, ossia $$x+y$$\  
\  
Utilizziamo come esemplio la regressione lineare semplice: $$ y\_i = \\beta\_0 + \\beta\_1 x\_i + \\varepsilon\_i \\qquad i = 1,\\dots,n$$ \  
\  
Definiamo gli stimatori per $\\beta\_0$ (`\\beta\_0`) e $\\beta\_1$ (`\\beta\_1`) come:\  
\  
- $\\hat\{\\beta\}\_0= \\bar\{y\} - \\hat\{\\beta\}\_1\\bar\{x\}$ (`$\\hat\{\\beta\}\_0= \\bar\{y\} - \\hat\{\\beta\}\_1\\bar\{x\}$`)\  
- $\\hat\{\\beta\}\_1 = \\frac\{\\sigma\_\{XY\}\}\{\\sigma\_\{XX\}\}$ (`$\\hat\{\\beta\}\_1 = \\frac\{\\sigma\_\{XY\}\}\{\\sigma\_\{XX\}\}$`)\  
\  
\  
## Header\  
Consideriamo ora le impostazioni di header per un [documento html](<http://rmarkdown.rstudio.com/html_document_format.html>).\  
\  
```\  
---\  
title: "Introduzione a R-Markdown"\  
subtitle: "Ovvero: come scrivere documenti in R"\  
author: "Tullia Padellini"\  
date: "21 Aprile 2017"\  
output: html\_document\  
---\  
```\  
\  
Possiamo aggiungere una `table of contents` con l'opzione `toc: true` (\_\_ATTENZIONE\_\_: questo non \'e8 codice `R` per questo `true` \'e8 scritto minuscolo e l'assegnazione \'e8 fatta tramite due punti e non uguale)\  
\  
```\  
---\  
title: "Introduzione a R-Markdown"\  
subtitle: "Ovvero: come scrivere documenti in R"\  
author: "Tullia Padellini"\  
date: "21 Aprile 2017"\  
output: \  
html\_document:\  
toc: true\  
toc\_float: true\  
---\  
```\  
\  
Piccola selezione di opzioni YAML utili: \  
```\  
---\  
title: "Introduzione a R-Markdown"\  
subtitle: "Ovvero: come scrivere documenti in R"\  
author: "Tullia Padellini"\  
date: "21 Aprile 2017"\  
output: \  
html\_document:\  
toc: true\  
toc\_float: true\  
number\_sections: true\  
code\_folding: hide\  
theme: bootstrap\  
highlight: tango\  
fig\_width: 10\  
fig\_height: 3\  
---\  
```\  
\  
\* `number\_sections: true` permette di numerare le sezioni\  
\* `code\_folding: hide` compatta i box di codice\  
\* `theme: bootstrap` specifica il tema `bootstrap` (tema di default). Altre opzioni disponibili sono: "cerulean", "journal", "flatly", "readable", "spacelab", "united", "cosmo", "lumen", "paper", "sandstone", "simplex", e "yeti".\  
\* `highlight: tango` imposta come colore di default il colore `tango`. Altre opzioni disponibili sono: "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock", e "textmate".\  
\* `fig\_width` e `fig\_height` sono invece i valori per le dimensioni delle figure nei plot `R`, che possono quindi essere specificati qui invece che volta per volta. \  
\  
\  
Ci sono poi diversi pacchetti per personalizzare il formato di output, in particolare potete usare:\  
\  
- rmdformats o tufte (che contengono diversi modelli per formato `html`)\  
- rticles (che contiene diversi modelli per articoli in stile `LaTeX`)\  
- [il vostro](<http://rmarkdown.rstudio.com/developer_custom_formats.html>)\  
\  
\  
\  
\  
## Codice R\  
\  
In un documento R-markdown istruzioni di codice vengono inserite in blocchi delimitati da tre apici ` ``` `. Un generico blocco di codice ha la seguente forma:\  
\  
`\  
```\{ opzioni \}\  
`\  
\  
`\  
istruzioni \  
`\  
\  
`\  
```\  
`\  
\  
Se poi vogliamo che questo codice sia eseguito dobbiamo aggiungere informazioni. Per prima cosa dobbiamo specificare che il codice che abbiamo scritto fa riferimento al linguaggio `R`, e per farlo specifichiamo `r` tra le opzioni. \  
\  
Indichiamo quindi con ` ```\{r\} ` l'inizio del blocco di codice R e con ` ``` ` la fine. \  
\  
Naturalmente questa \'e8 solo una delle tante possibili opzioni ([qui](<https://yihui.name/knitr/options/>) potete trovare l'elenco completo).\  
\  
```\{r\}\  
# Questo \'e8 un blocco di codice R\  
```\  
\  
Analizziamo i dati del file `Circonferenza-Cranica.csv`, iniziando (come al solito) dall'importazione dei dati in `R`.\  
\  
\_\_ATTENZIONE\_\_: \  
\  
\* Se i dati che volete caricare sono contenuti in un file di testo (ad esempio un file `.csv` o `.txt`) la funzione da usare \'e8 `read.table()`. \  
\* Se i dati che volete caricare sono contenuti in un workspace `R`, ossia in un file `.RData` la funzione da usare \'e8 `load()`.\  
\  
```\{r\}\  
dati=read.csv("Circonferenza-Cranica.csv", header=T, sep=";")\  
```\  
\  
\  
Possiamo utilizzare tutte le funzioni che gi\'e0 conosciamo per questo dataframe; con le opzioni di default nel Markdown appariranno due blocchi, il primo contente il codice ed il secondo contente il corrispettivo output. Consideriamo ad esempio la funzione `str()`:\  
```\{r\}\  
str(dati)\  
names(dati)\  
```\  
\  
Analogamente per le altre funzioni. \  
```\{r\}\  
names(dati)\  
summary(dati)\  
```\  
\  
```\{r\}\  
n=nrow(dati)\  
```\  
\  
Per mostrare il codice senza mandarlo in esecuzione, possiamo utilizzare l'opzione `eval` (che di default \'e8 uguale a `TRUE`), ossia specificare `\{r, eval=FALSE\}` all'inzio del blocco di codice. \  
```\{r, eval=FALSE\}\  
str(dati)\  
```\  
\  
In modo simile, grazie all'opzione `echo` possiamo anche mostrare solo l'output senza il codice che lo ha generato. In particolare lo stesso blocco di codice, preceduto da `\{r, echo=FALSE\}` produrr\'e0 qualcosa del tipo:\  
\  
```\{r, echo=FALSE\}\  
str(dati)\  
```\  
\  
Il dataset \*Circonferenza Cranica\* \'e8 composto da $6$ variabili:\  
\  
- `headcirc`: circonferenza cranica\  
- `lenght`: lunghezza / altezza del bambino\  
- `gestage`: eta' gestazionale \  
- `birthwt`: peso alla nascita\  
- `momage`: eta' della madre al momento del parto\  
- `toxemia`: presenza/assenza di tossemia della madre durante la gravidanza\  
\  
Possiamo codificare la tossemia come `factor`:\  
```\{r\}\  
dati$toxemia=factor(dati$toxemia)\  
```\  
\  
### Grafici\  
Possiamo inserire dei grafici nel markdown:\  
```\{r\}\  
boxplot(dati$headcirc ~ dati$toxemia, col = c(rgb(.3,.2,.3,.4),rgb(.3,0,.3,.4)),\  
range = 0)\  
```\  
\  
Come potete vedere, il grafico non \'e8 centrato ma \'e8 allineato a sinistra. Per cambiare questa impostazione, utilizziamo l'opzione `fig.align` da inserire all'inizio del blocco; in particolare con `\{r, fig.align='center'\}` scegliamo l'allineamento centrato. Come si pu\'f2 facilmente intuire `fig.align='right'` corrisponde all'allineamento a destra e `fig.align='left'` a quello a sinistra (valore di default).\  
\  
```\{r, fig.align='center'\}\  
boxplot(dati$headcirc ~ dati$toxemia, col = rgb(.3,.2,.3,.4),\  
range = 0)\  
```\  
Anche se adesso il grafico \'e8 centrato, non \'e8 possibile leggere tutte le etichette perch\'e8 \'e8 troppo stretto. Possiamo cambiare altezza e larghezza dei grafici aggiungendo le opzioni `fig.width` e `fig.height`. In particolare `\{r, fig.width=15, fig.height=7.5, fig.align='center'\}` produce un grafico centrato di larghezza 15 cm ed altezza 7.5 cm. \  
\  
```\{r, fig.width=15, fig.height=7.5, fig.align='center'\}\  
boxplot(dati$headcirc ~ dati$toxemia, col = rgb(.3,.2,.3,.4),\  
range = 0)\  
```\  
\  
Naturalmente possiamo mettere un numero arbitrario di grafici nella finestra attiva grazie al comando `par(mfrow=c())`. \  
```\{r, fig.align='center', fig.width=15\}\  
par(mfrow=c(1,2))\  
boxplot(dati$headcirc, range = 0, pch=20, col=rgb(.3,.2,.3,.4), main ="Boxplot")\  
hist(dati$headcirc, col=rgb(.3,.2,.3,.4), main="Istogramma", xlab="", ylab="")\  
abline(v=median(dati$headcirc),col=rgb(.9,.2,.3,.4), lwd=3)\  
abline(v=quantile(dati$headcirc, c(0.25, 0.75)),col=rgb(.3,.9,.3,.4), lwd=3)\  
par(mfrow=c(1,1))\  
```\  
\  
\  
\  
### Tabelle\  
\  
Consideriamo ora le variabili $X$=`dati$toxemia` e $Y$=`dati$gestage`. Avete gi\'e0 visto che con la funzione `table()` possiamo ottenere la tabella di frequenze doppie come output di `R`:\  
```\{r\}\  
tab = table(dati$toxemia, dati$gestage)\  
tab\  
```\  
\  
Se vogliamo scrivere la stessa tabella utilizzando la sintassi di markdown, possiamo farlo nel seguente modo.\  
```\  
| | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |\  
| ----- |:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|\  
| 0 | 2 | 2 | 7 | 5 | 12 | 9 | 18 | 10 | 8 | 3 | 3 | 0 | 0 |\  
| 1 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 3 | 3 | 2 | 5 | 1 | 1 |\  
```\  
\  
\  
| | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |\  
| ----- |:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|:--:|\  
| 0 | 2 | 2 | 7 | 5 | 12 | 9 | 18 | 10 | 8 | 3 | 3 | 0 | 0 |\  
| 1 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 3 | 3 | 2 | 5 | 1 | 1 |\  
\  
\  
Pu\'f2 essere comodo creare una tabella `html` direttamente dall'output di `R`, senza dover copiare a mano tutti i dati; il modo pi\'f9 semplice di farlo \'e8 attraverso la funzione `kable()`, disponibile nel pacchetto `knitr`\  
\  
\_\_ATTENZIONE\_\_ per generare correttamente le tabelle, dobbiamo specificare l'opzione `results='asis'` all'inizio del chunck di codice.\  
```\{r, results='asis', eval=FALSE\}\  
library(knitr)\  
kable(tab, digits = 2, align = c(rep("l", 4), rep("c", 4), rep("r", 4)))\  
```\  
\  
Esistono diversi altri pacchetti che permettono di farlo, tra cui:\  
\  
\* [xtable](<https://cran.r-project.org/web/packages/xtable/>)\  
\* [stargazer](<https://cran.r-project.org/web/packages/stargazer/>)\  
\* [pander](<http://rapporter.github.io/pander/>)\  
\* [tables](<https://cran.r-project.org/web/packages/tables/>)\  
\* [ascii](<http://eusebe.github.io/ascii/>)\  
\* etc.\  
\  
Consideriamo ad esempio `xtables`.\  
\  
```\{r, results='asis'\}\  
library(xtable)\  
tt = xtable(tab, caption = 'Tabella di Frequenze', auto=TRUE)\  
print.xtable(tt, type="html",floating.environment = 'margintable' )\  
```\  
\  
Possiamo costruire anche tabelle pi\'f9 elaborate, ad esempio consideriamo la tabella delle frequenze relative rispetto al totale di riga:\  
```\{r\}\  
prop.table(tab,1) \  
\  
# per controllare che siano effettivamente frequenze relative possiamo verificare che la somma per riga sia 1\  
addmargins(prop.table(tab,1),2)\  
```\  
\  
```\{r, results='asis'\}\  
tt= xtable(prop.table(tab,1))\  
print(tt, type="html")\  
```\  
\  
Costruiamo poi le distribuzioni condizionate percentuali di $Y$ rispetto ad $X=x\_i$\  
```\{r\}\  
prop.table(tab,1)\*100\  
addmargins(prop.table(tab,1)\*100,2)\  
```\  
\  
```\{r, results='asis'\}\  
print(xtable(prop.table(tab,1)\*100), type='html')\  
```\  
\  
e le distribuzioni condizionate relative della $X$ rispetto a $Y = y\_j$\  
```\{r\}\  
prop.table(tab,2)\*100\  
addmargins(prop.table(tab,2)\*100,1)\  
```\  
\  
```\{r, results='asis'\}\  
print(xtable(prop.table(tab,2)\*100), type='html')\  
```\  
\  
\  
\  
```\{r\}\  
marg.X=margin.table(tab,1) \  
marg.Y=margin.table(tab,2) \  
```\  
\  
```\{r, results='asis'\}\  
print(xtable(marg.X), type='html')\  
print(xtable(marg.Y), type='html')\  
```\  
\  
}