Compte rendu TP2 - Régression linéaire, probabilités, intervalle de confiance

Laroussi Labid Bachri, M1 BBS

13-09-2025

Contents

Dans ce TP, nous allons etudier la regression linéaire, les calculs de probabilité, des calculs bases sur la loi binomiale et finalement les intervalles de confiance.

Dans ce tp, on utilisera la libraire here pour la gestion des chemins relatifs.

Régression linéaire

Nous allons découvrir sur la regression linéaire en travaillant sur le génome d'une bacterie.

```
genomes = read.table(params$fichier_bacteries , sep = params$separateur_bacterias , header = TRUE)
names(genomes)
```

```
## [1] "Genome_size" "ORF_number"
```

```r
with(genomes , plot(Genome_size , ORF_number , pch=16 , ylab="Nombre d'ORF" , xlab = "Taille du génome

# L'unité de mesure est de l'ordre des milions de paires de bases.

# On peut calculer la relation entre ces 2 variables mais aussi le coefficient de Pearson.

cov_xy = with(genomes, cov(Genome_size , ORF_number))
r_xy = with(genomes, cor(Genome_size , ORF_number))
cat(sprintf("Covariance = %.0f ; r (Pearson) = %.2f\n", cov_xy, r_xy))
```

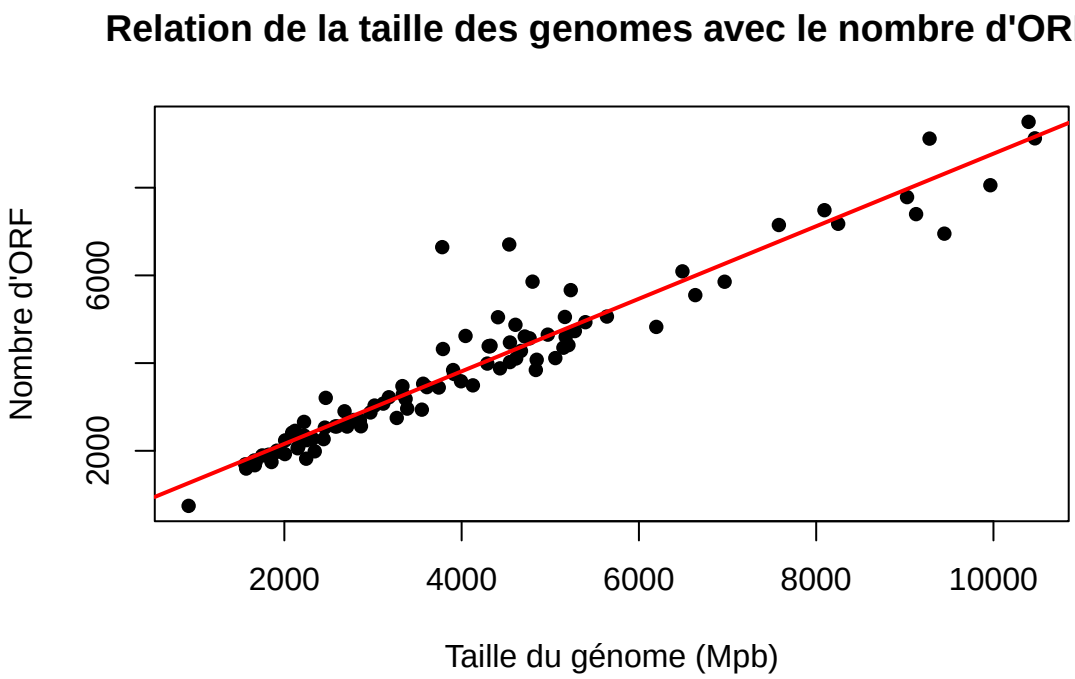
```
## Covariance = 4005883 ; r (Pearson) = 0.96
```

```r
#On observe une valeur de covariance de 4005883 et un coefficient de correlation de Pearson de 0.96. On

ligneregression= lm(ORF_number~Genome_size, data = genomes) # Nous n'avons utilisé pas attach donc faut

# Ajout de la droite de regréssion

relation_Genome_size_nb_ORF = with(genomes , plot(Genome_size , ORF_number , pch=16 , ylab="Nombre d'ORF
abline(ligneregression , col="red", lwd=2)
```




```r
summary(ligneregression)
```

```
##
## Call:
## lm(formula = ORF_number ~ Genome_size, data = genomes)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1365.3  -222.2  -101.9   149.4  3013.0
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 503.29112  117.28721   4.291 4.21e-05 ***
## Genome_size   0.82723    0.02561  32.304  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 557.8 on 97 degrees of freedom
## Multiple R-squared:  0.915,  Adjusted R-squared:  0.9141
## F-statistic:  1044 on 1 and 97 DF,  p-value: < 2.2e-16
```

```
relation_Genome_size_nb_ORF = function() {
with(genomes , {plot(Genome_size , ORF_number , pch=16 ,
                     ylab="Nombre d'ORF" ,
                     xlab = "Taille du génome (Mpb)" ,
                     main="Relation de la taille des genomes avec le nombre d'ORF. " )
abline(ligneregression , col="red", lwd=2)
})
  }

relation_Genome_size_nb_ORF()

outfile_bacterias = here("figure_tp2" , "relation_Genome_size_nb_ORF.jpg")
jpeg(outfile_bacterias)
relation_Genome_size_nb_ORF()
dev.off()
```

```
## pdf
##   2
```

Coefficients de la droite

Les coefficients de la droite de régression seront a = 0.8272, b = 503.39. On peut donc en deduire l'équation y = 0.8272x + 503.39

La droite de régression est en outil très intéressant pour la prediction et pouvoir estimer des valeurs en lisant direcement sur la figure.

Calculs de probabilité

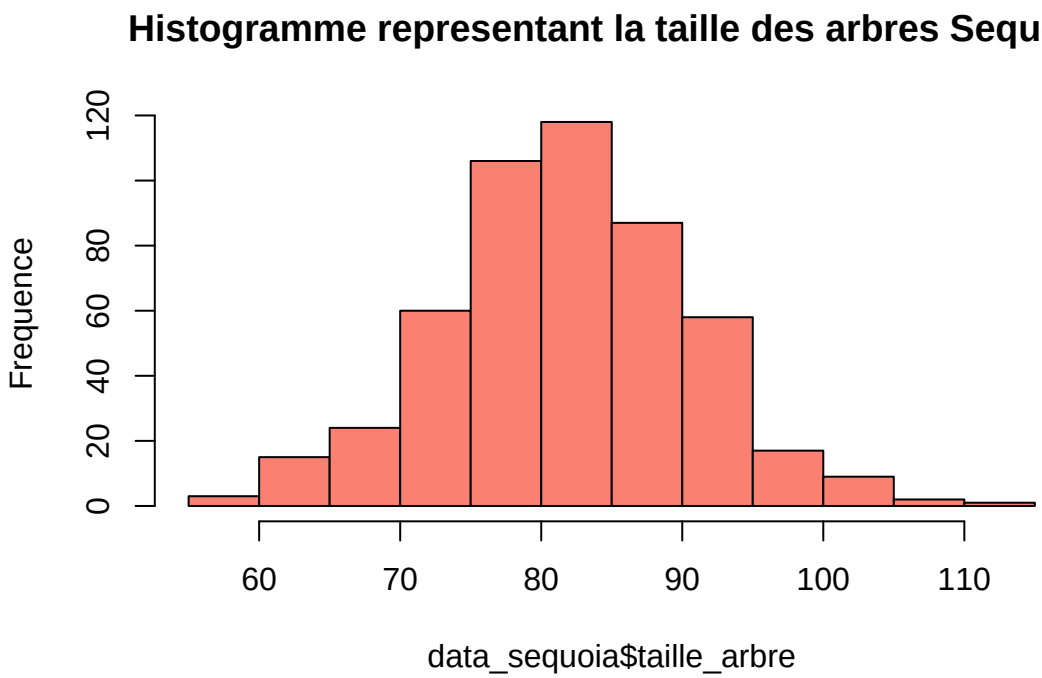
Nous allons maintenant travailler sur les calculs basés sur des données. L'analyse porte sur la taille des arbres sequoia.

```
data_sequoia = read.table(params$fichier_sequoia , sep = params$separateur_sequoia , header = TRUE)

names(data_sequoia)
```

```
## [1] "taille_arbre"
```

```
hist(data_sequoia$taille_arbre,
     main="Histogramme representant la taille des arbres Sequoia",
     ylab='Frequence',
     col='salmon' )
```



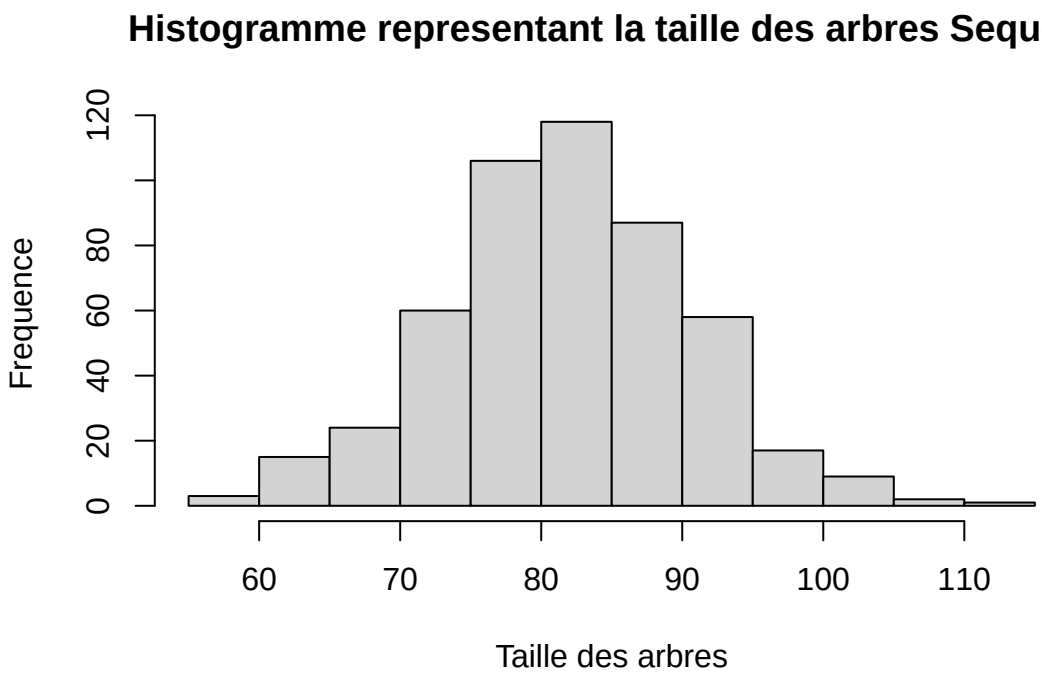

```
mean_taille_sequoia = mean(data_sequoia$taille_arbre)
ecart_type_sequoia = sd(data_sequoia$taille_arbre)

cat("La taille moyenne est de" , mean_taille_sequoia , " et l'écart-type est" , ecart_type_sequoia
    )
```

```
## La taille moyenne est de 82.268  et l'écart-type est 8.746037
```

```
histogramme_taille_arbre = function() { hist(data_sequoia$taille_arbre , main="Histogramme representant

histogramme_taille_arbre()
```




```
outfile_seq = here("figure_tp2" , "histogramme_taille_arbre.jpg")
jpeg(outfile_seq)
histogramme_taille_arbre()
dev.off()
```

```
## pdf
##   2
```

```
if ( file.exists(outfile_seq)) {
  cat("Le fichier a bien ete generée.
      " , outfile_seq )
  } # Utilisation de if pour envoyer un output qui confirme l'existence de notre fichier.
```

```
## Le fichier a bien ete generée.
##        /Users/bachri/Documents/M1_BBS/S7/tdb/tp_tdb/figure_tp2/histogramme_taille_arbre.jpg
```

```
## Quel est la probabilité qu'un arbre mesure 80 m ?


proba_arbre_80m = length(which(data_sequoia$taille_arbre==80))/length(data_sequoia$taille_arbre)

cat("La probabilité qu'un arbre mesure 80m est de", proba_arbre_80m)
```

```
## La probabilité qu'un arbre mesure 80m est de 0.044
```

```
## Quel est la probabilité qu'un arbre mesure 100 m ?


proba_arbre_100m = length(data_sequoia$taille_arbre[data_sequoia$taille_arbre>100])/length(data_sequoia$

cat("La probabilité qu'un arbre mesure 100m est de", proba_arbre_100m)
```

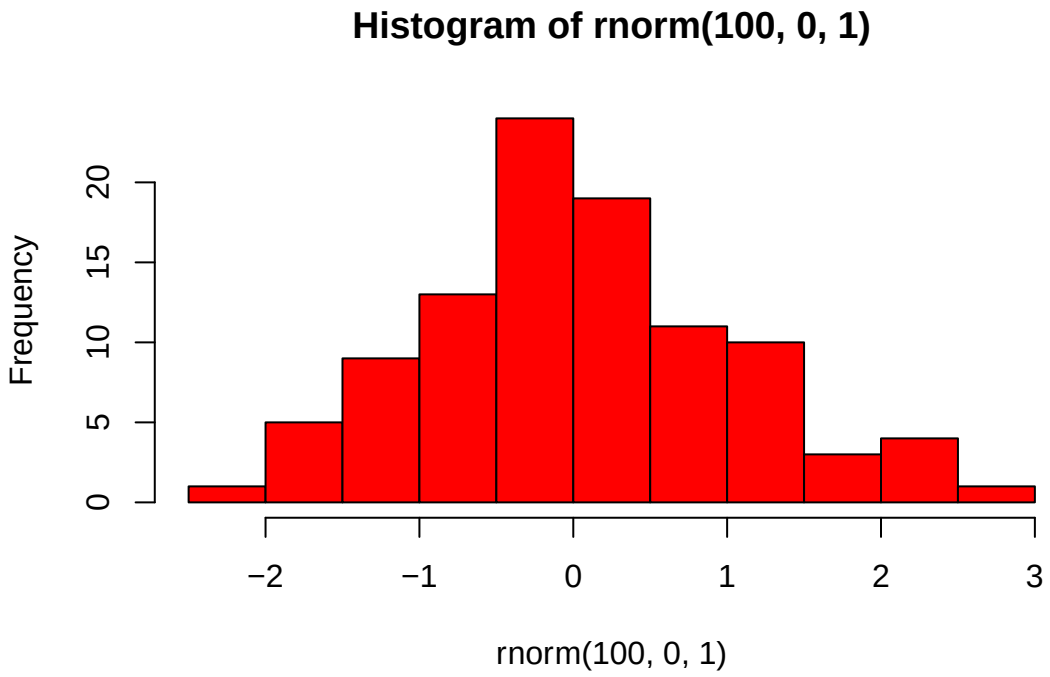
```
## La probabilité qu'un arbre mesure 100m est de 0.024
```

```
## Calcul basés sur une loi centrée réduite N(0,1)


dnorm(0,0,1) # On a ici x=0, mean=0 et sd=1 respectivement. Ici , x est une variable aleatoire, mean co
```

```
## [1] 0.3989423
```

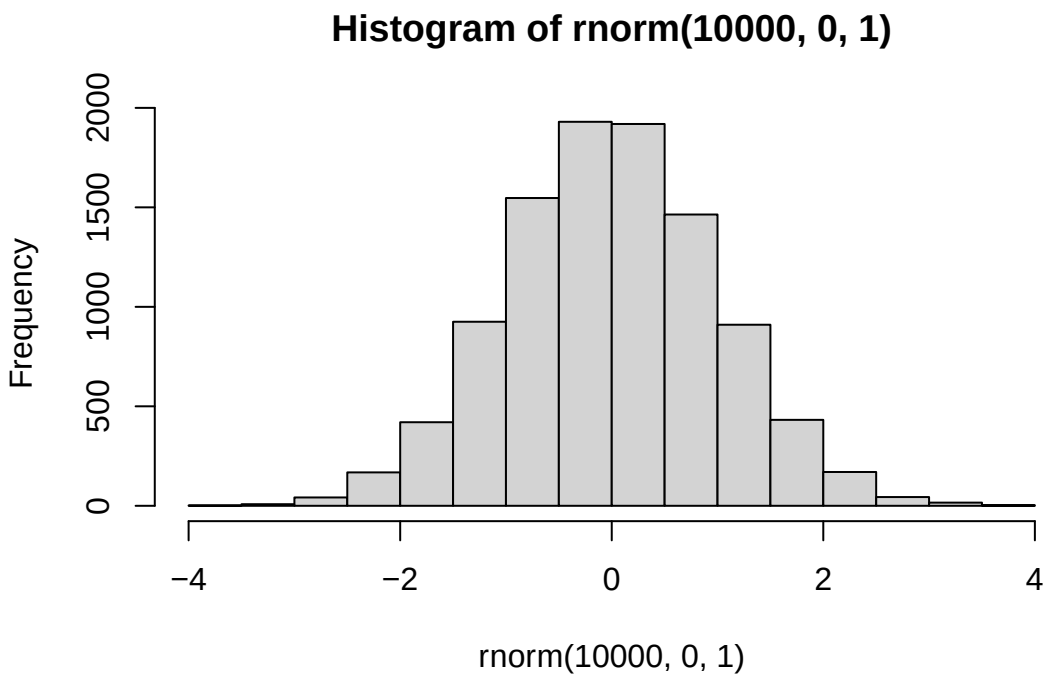
```
hist(rnorm(100,0,1), col='red')
```




```
# En augmentant n, on va tendre vers une figure de plus en plus symetrique centrée normale.
pnorm(0,0,1)
```

```
## [1] 0.5
```

```
hist(rnorm(10000,0,1))
```




```
# On va calculer la mean et écart type des tailles des séquoia pour calculer la probabilité qu'u arbre
```

```
calcul_arbre80m = round(dnorm(80,mean_taille_sequoia , ecart_type_sequoia ) , 3)
calcul_arbre80m
```

```
## [1] 0.044
```

```
calcul_arbre100m = round(pnorm(100,mean_taille_sequoia, ecart_type_sequoia, lower.tail = FALSE),3)
calcul_arbre100m
```

```
## [1] 0.021
```

L'option lower.tail = F est utilisé pour trouver la valeur que la probabilité de la variable aleatoire normal soit inferieur ou superieure a q. lower.tail= TRUE pour valeurs inferieurs. Par défaut, lower.tail=TRUE

Ici, il faut utiliser pnorm pour avoir la somme des probabilités des arbres qui font plus de 100m. Cumulative.

```
## La probabilité qu'un arbre fasse 80m selon la loi normal est de 0.044
```

```
## Le calcul des probabilités qu'un arbre mesure 100m ou plus est de 0.021 . Cette valeur est legeremen
```

Ces differences sont du à l'échantillonage donc on aura toujours une différence.

Calculs basés sur une loi binomaile B(n,p)

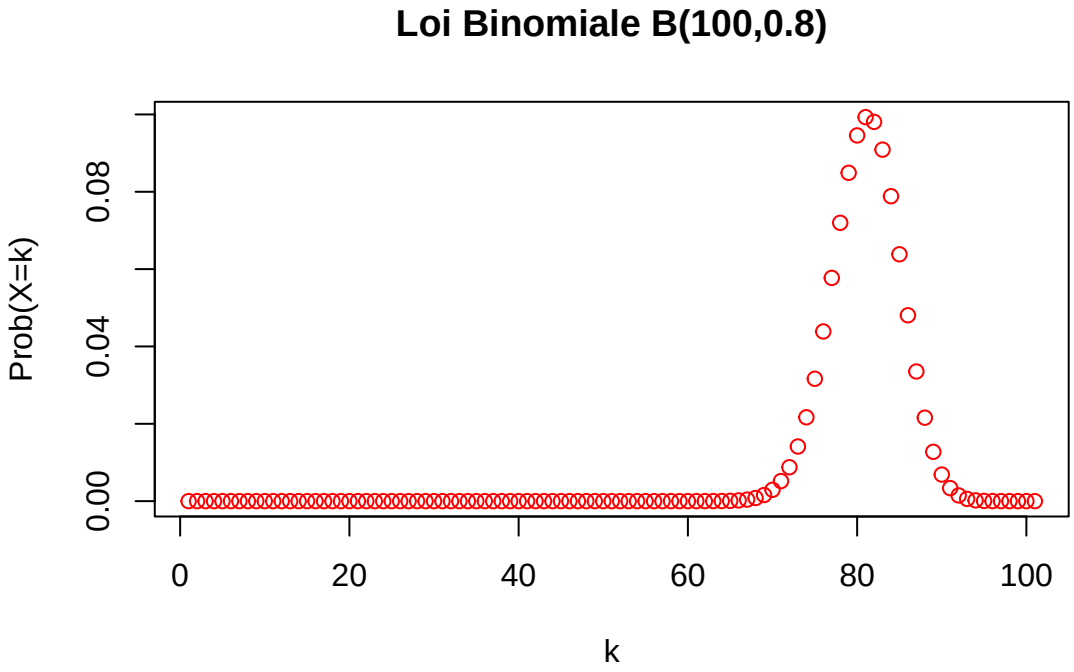
```r
n=100
p=0.8
proba_80_graines_germent = dbinom(80,n,prob=p)
proba_au_max80_graines_germent = pbinom(80,n,prob=p)
proba_plus_de_80_graines_germent = pbinom(80,n,prob=p , lower.tail=F)
cat("La probabilité que 80 graines germent est de" ,proba_80_graines_germent)
```

```
## La probabilité que 80 graines germent est de 0.09930021
```

```
## La probabilité que au maximum 80 graines germent est de 0.5398386
```

```
## La probabilité que plus de 80 graines germent est de 0.4601614
```

```
## D'apres notre n et p, on devrait s'attendre en mean a 80
```




```
## pdf
##   2
```

```
## Le fichier a bien ete genérée.  /Users/bachri/Documents/M1_BBS/S7/tdb/tp_tdb/figure_tp2/figure_loi_b
```

On aura ici des points et pas une droite car la loi binomiale est une loi discrete et non continue.

Estimer un intervalle de confiance (IC) d'un paramètre populationnel (μ, p,...) à partir d'un échantillon

```r
data_tomatos = read.table(params$fichier_tomatos , sep = params$separateur_tomatos , header=T)
names(data_tomatos)
```

```
## [1] "poids_tomate"
```

```r
mean_tomatos = mean(data_tomatos$poids_tomate)
sd_tomatos = sd(data_tomatos$poids_tomate)
n_tomatos = length(data_tomatos$poids_tomate)


# Intervalle de confiance de la mean théorique μ de la popualtion, à partir de la mean m calculé sur un
# Intervalle de confiance a 95%

error = qt(0.975, df=n_tomatos-1)*sd_tomatos/sqrt(n_tomatos)
left = mean_tomatos - error
right = mean_tomatos + error
left;right
```

```
## [1] 11.82531
```

```
## [1] 12.41069
```

```
## L'intervalle de confiance a 95% sera compris entre 11.82531 et 12.41069 . On peut estimer que la mea
```

On peut augmenter l'intervalle de confiance mais on aura des valeurs plus importantes.

```r
error_99 = qt(0.995, df=n_tomatos-1)*sd_tomatos/sqrt(n_tomatos)
left_99_tomatos = mean_tomatos - error_99
right_99_tomatos = mean_tomatos + error_99
left_99_tomatos;right_99_tomatos
```

```
## [1] 11.72767
```

```
## [1] 12.50833
```

```
## L'intervalle de confiance a 95% sera compris entre 11.72767 et 12.50833 . On peut estimer que la mea
```

Cet intervalle pourra etre plus petite en augmentant notre nombre de tomatos.

```r
frequence_red_apples = 0.4 # fréquence de pommes rouges dans l'échantillon
n = 125 # taille de l'échantillon

error_red_apple = qnorm(0.975)*sqrt(frequence_red_apples*(1-frequence_red_apples)/n)

left_apples = frequence_red_apples - error_red_apple
right_apples = frequence_red_apples + error_red_apple
left_apples;right_apples
```

```
## [1] 0.3141187
```

```
## [1] 0.4858813
```

```r
cat("L'intervalle ayant 95% de chance d'obtenir la vrai proportion p de la popualtion est entre" , left
```

```
## L'intervalle ayant 95% de chance d'obtenir la vrai proportion p de la popualtion est entre 0.3141187
```

Sources utiles

- https://www.datamentor.io/r-programming/if-else-statement (if else/)
- https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/sprintf (sprintf)
- [https://stat.ethz.ch/R-manual/R-devel/library/base/html/sprintf.html] (https://stat.ethz.ch/R-manual/R-devel/library/base/html/sprintf.html) (sprintf)
- https://yihui.org/knitr/options/ (Options knitr)