Statistical Learning

Due Monday, May 15 on Moodle

Homework-01

General Instructions

- You can use *any* programming language you want, as long as your work is runnable/correct/readable. Two examples:
  - **In `R`**: it would be nice to upload a well-edited and working `R Markdown` file (`.rmd`) + its `html` output.
  - **In `Python`**: it would be nice to upload a well-edited and working `notebook` (or similia).

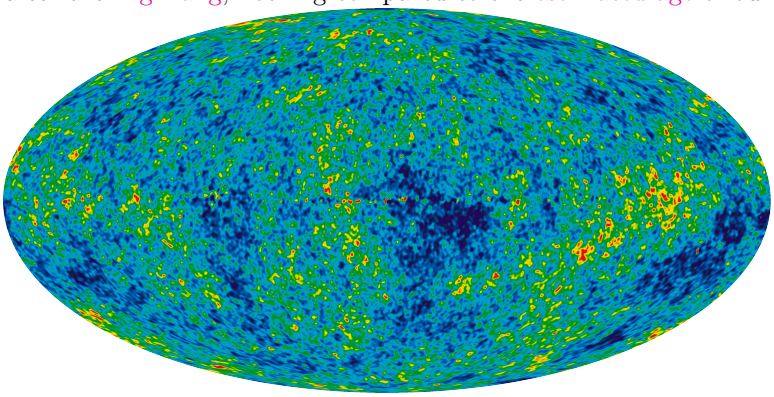
In case of `R`

If you go for `R`, to be sure that everything is working, start `RStudio` and create an empty project called `HW1`. Now open a new `R Markdown` file (`File > New File > R Markdown...`); set the output to `HTML mode`, press `OK` and then click on `Knit HTML`. This should produce a `html`. You can now start editing this file to produce your homework submission.

- For more info on `R Markdown`, check the support webpage: R Markdown from RStudio.
- For more info on how to write math formulas in LaTex: Wikibooks.

The Data

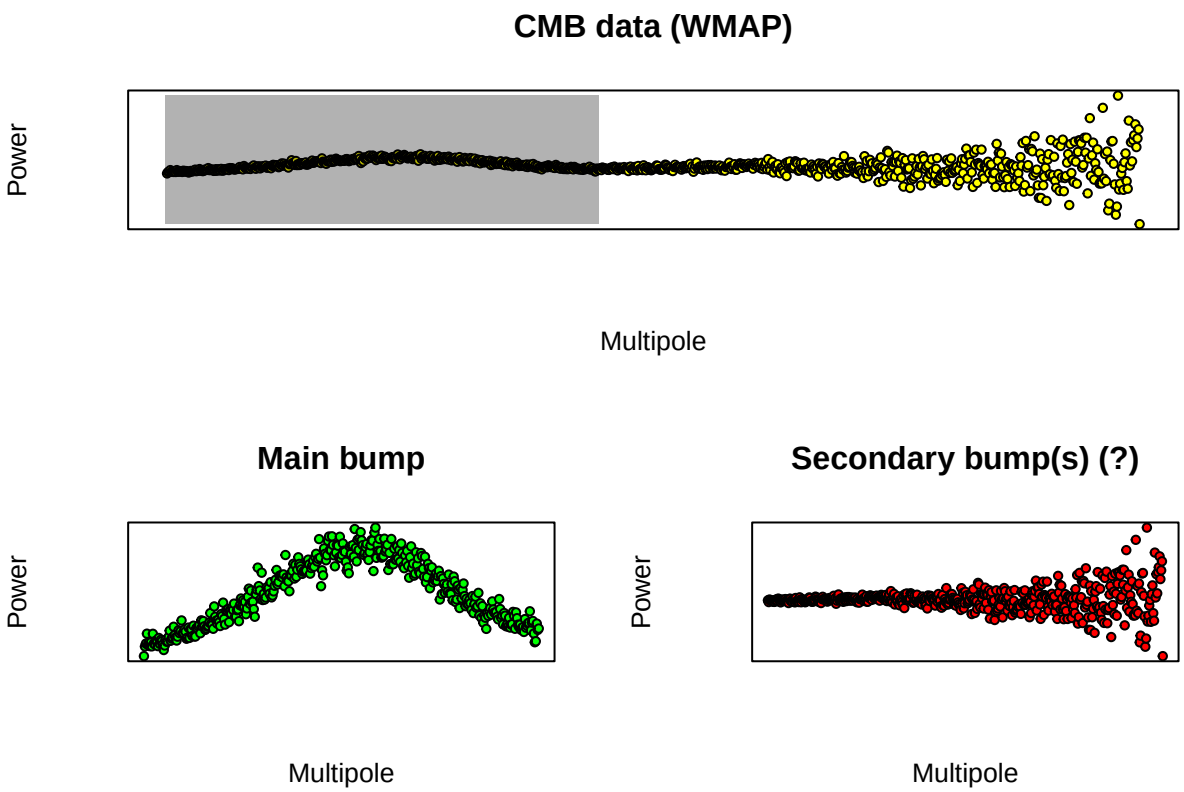
For this homework you'll be working on the WMAP data. We are talking about a snapshot of our universe in its *infancy*, something like 379,000 years after the Big Bang, nothing compared to the estimated age of our universe[1].



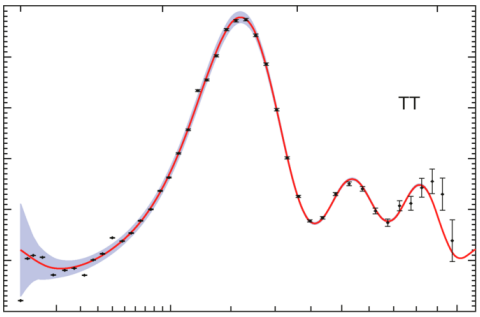
The map above was taken by the Wilkinson Microwave Anisotropy Probe (WMAP) and shows differences across the sky in the temperature of the cosmic microwave background (CMB), the radiant heat remaining from the Big Bang. The average temperature is 2.73 degrees above absolute zero but the temperature is not constant across the sky. The fluctuations in the temperature map provide information about the early universe. Indeed, as the universe expanded, there was a tug of war between the force of expansion and contraction due to gravity. This caused acoustic waves in the hot gas, which is why there are temperature fluctuations.

The strength of the temperature fluctuations $f(x)$ at each frequency (or multipole) x is called the **power spectrum** and this power spectrum can be used by cosmologists to answer cosmological questions. For example, the relative abundance of different constituents of the universe (such as baryons and dark matter) corresponds to **peaks** in the power spectrum. Through a complicated procedure (not described here), the temperature map can be reduced to the following scatterplot of power versus frequency:

[1] ...something like 14 *billion* years!



The horizontal axis is the multipole moment, essentially the frequency of fluctuations in the temperature field of the CMB. The vertical axis is the power or strength of the fluctuations at each frequency. The top plot shows the full data set. The two bottom plots zoom in the first 400 and the next 500 data points. The 1^{st} peak, around $x \approx 200$, is obvious. But many "official" fit (see **Fig. 2** in that page) show also a 2^{nd} and 3^{rd} peak further to the right...



Part I: Polynomials are good...

...but smooth piecewise polynomials (a.k.a. splines) are better!

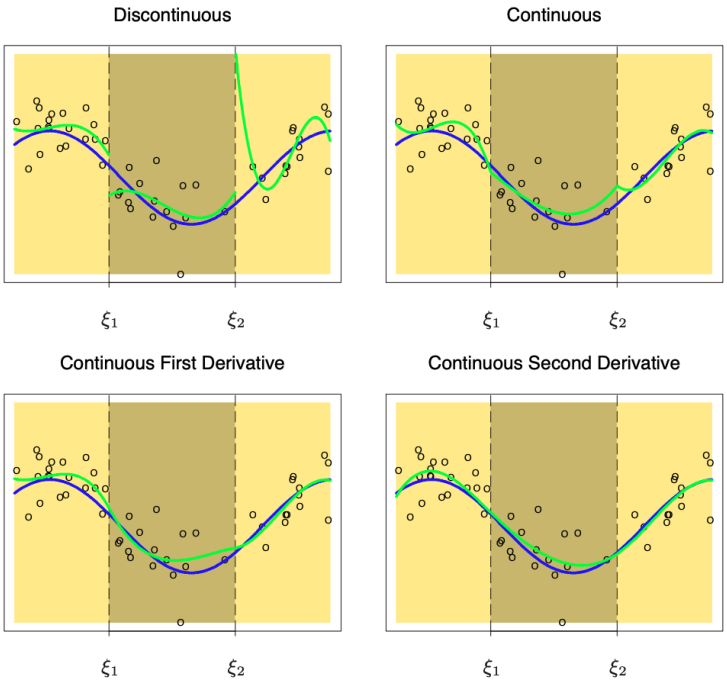
In our initial supervised toy example we tried to recover from samples a *non-linear* (in the original 1-dimensional feature-space) function using a linear model in a higher-dimensional transformed feature-space (polynomials of degree d) Now, polynomials are fine, but they are *global* objects (they span the entire real-line in principle) and may not be able to capture *local* structures of the target function (also, Mairhuber-Curtis pushed for more "data-embracing" solutions)

RECIPE FOR A SOLUTION: 1. take a polynomial, 2. break it down is small (almost) free-to-move chunks, 3. shake a bit, 4. glue them back together adding some regularity constraint (continuity, differentiability, etc) as needed... a *spline* is born...

MORE FORMALLY: any d^{th}-order spline $f(\cdot)$ is a **piecewise polynomial function** of degree d that is continuous and has continuous derivatives of orders $\{1, \dots, d-1\}$ at the so called *knot points*. Specifically, how do we build a generic d^{th}-order spline $f(\cdot)$? We start from a bunch of points, say q, that we call *knots* $\xi_1 < \cdots < \xi_q$, and we then ask the following:

1. $f(\cdot)$ is *some* polynomial of degree d on each of the intervals: $(-\infty, \xi_1], [\xi_1, \xi_2], [\xi_2, \xi_3], \dots, [\xi_q, +\infty)$;
2. its j^{th} derivative $f^{(j)}(\cdot)$ is continuous at $\{\xi_1, \dots, \xi_q\}$ for each $j \in \{0, 1, \dots, d-1\}$.

The figure below from Chapter 5 of ELS illustrates the effects of enforcing continuity at the knots, across various orders of the derivative, for a cubic piecewise polynomial.



Splines have some amazing properties, and they have been a topic of interest among statisticians and mathematicians for a very long time (classic vs recent). **But**, given a set of points $\xi_1 < \xi_2 < \cdots < \xi_q$, is there a quick-and-dirty way to describe/generate the whole set of d^{th}-order spline functions over those q knots? The easiest one (**not the best!**), is to start from **truncated power functions** $\mathcal{G}_{d,q} = \{g_1(x), \dots g_{d+1}(x), g_{(d+1)+1}(x), \dots, g_{(d+1)+q}(x)\}$, defined as

$$\{g_1(x) = 1, g_2(x) = x, \dots, g_{d+1}(x) = x^d\} \text{ and } \{g_{(d+1)+j}(x) = (x - \xi_j)_+^d\}_{j=1}^q \text{ where } (x)_+ = \max\{0, x\}.$$

Then, if $f(\cdot)$ is a d^{th}-order spline with knots $\{\xi_1, \dots, \xi_q\}$ you can show it can be obtained as a *linear combinations* over $\mathcal{G}_{d,q}$

$$f(x) = \sum_{j=1}^{(d+1)+q} \beta_j \cdot g_j(x), \text{ for some set of coefficients } \boldsymbol{\beta} = \left[\beta_1, \dots, \beta_{d+1}, \beta_{(d+1)+1}, \dots \beta_{(d+1)+q}\right]^{\mathtt{T}}.$$

IDEA: let's perform regression on splines instead of polynomials! In other words, as in our initial toy example, given inputs $\boldsymbol{x} = \{x_1, \dots, x_n\}$ and responses $\boldsymbol{y} = \{y_1, \dots, y_n\}$, consider fitting functions $f(\cdot)$ that are d^{th}-order splines with knots at some chosen locations, typically the first q quantiles of $\boldsymbol{x}$ $\rightsquigarrow$ this method is dubbed **regression splines** and it is different from another famous technique called *smoothing splines* (we will talk about it later as an example of *(Mercer) kernel method*).

REMARK: here you have many tuning parameters (the degree d and the number+position of knots) to be selected *predictively* via `Cp` or some flavor of cross-validation (sample-splitting, k-fold `CV`, `LOOCV`, `GCV`). Although there is a large literature on knot selection for regression splines via greedy methods like recursive partitioning, here we will go for an easy-to-implement option.

$\rightsquigarrow$ Your job $\leftsquigarrow$

1. First of all, briefly explain to me why this idea is yet another manifestation of our "*be linear in transformed feature space*" mantra. Do you "perceive" any technical difference with the "*orthogonal series expansion*" point of view?
2. Develop your own implementation of the truncated power basis $\mathcal{G}_{d,q}$, and then plot a few elements with $d \in \{1, 3\}$ and $q \in \{3, 10\}$ equispaced knots in the open interval $(0, 1)$.
   Once you've done, use `ChatGPT` (or any other chat-bot capable of coding) to achieve the same task. Report the sequence of prompts you tried and the bot's replies. How would you evaluate the quality of this tiny bot-based experience?
3. Time to move to our little Kaggle-competition and its associated shared notebook (one per team, all your code and also comments must be available there). Before introducing the constraints under which you will work, let me list **the rules** of this toy-competition:
   - **The Task(s)**
     Main Task: Prediction problem || more details below

Secondary Task: Methods and results explanation/presentation/visualization (I'll personally score this second task)
Your *base* overall final score is composed as follow:

$$\text{SCORE} = \big(0.6 \times \text{Main Task}\big) + \big(0.4 \times \text{Secondary Task}\big)$$

The resulting score will be remapped in 0-30 and additional points will be assigned for originality/activity level.

- **Cheating**
  No code-leakage allowed.
  If two codes are judged to be too similar, their final scores will be cut in half.
  If three codes are judged to be too similar, their final scores will be cut into thirds, and so on...
  No appeal shall be made at any time.
- **Start-End**
  The competition will start today, Wednesday April 26 at 17:00 and will end on Monday May 15 at 22:00am
- **Availability**
  Right after the competition starts, the link to access the (private) competition will be sent to the team-leaders only by email (check your spam folder please).
- **Kaggle Notebooks**
  Right after the competition starts, I will create and share a Kaggle notebook (in the requested language) with the team-leaders of each group.
  It may take up to 10 mins to get it done, be patient.
  In case of troubles on my side, I may ask the team-leaders (again by email, check the spam!) to create a notebook on their own and share it with me as editor.
  These notebooks will be the only source of code/results/comments I will evaluate, nothing else will be considered.
- **Reproducibility**
  You can work on your personal computers (of course) but all your pre-processing/models/submissions must be made available and be entirely reproducible as a separate code-version on the assigned Kaggle notebook: one code-version for each submission.
  As a suggestion, label each code-version with a meaningful name like: `Sub_01`, `Sub_02` etc.
  Set your random generators seeds properly and submit solutions **only** if they come directly from your notebook.
- **Submissions Limits**
  The team-leader (and the team-leader only) can submit an entry.
  If you are organized as a **Kaggle-team**, this is not an issue.
  Maximum Daily Submission: 20
  Number of submissions eligible for the final private leaderboard: 2
  Notice: The code-versions related to these entries must be clearly labeled and easy to find on the Notebook.
- **Prize(s)**
  ... for the more active teams? Surely there will be a prize!

And now back to the **constraints** that all submissions must verify:

1. You **must** use truncated power basis $\mathcal{G}_{d,q}$ to implement regression splines from scratch on the `wmap` data.
   The main point is to handmade the $n \times (d+1) + q$ design matrix $\mathbb{X}$ having generic entry

   $$\mathbb{X}[i,j] = g_j(x_i) \;\text{ for }\; i \in \{1,\dots,n\} \;\text{ and }\; j \in \big\{1,\dots,(d+1)+q\big\}.$$

   You can then use (penalized) least squares of any kind (your model, your choice) to get the optimal coefficients

   $$\widehat{\boldsymbol{\beta}} = \big[\widehat{\beta}_1,\dots,\widehat{\beta}_{d+1},\widehat{\beta}_{(d+1)+1},\dots,\widehat{\beta}_{(d+1)+q}\big]^{\intercal},$$

   which then leaves you with the fitted *regression spline* model: $\widehat{f}(x) = \sum_{j=1}^{(d+1)+q} \widehat{\beta}_j\, g_j(x)$.
2. You need to select the best (d,q)-combination via a suitable predictive criterion. Here's what you'll do:
   - Consider $d \in \{1,3\}$, that is linear and cubic-splines at least, and knots positioned on q-equispaced locations (within the x-range) $\rightsquigarrow$ grid-search between a min value $q_{\texttt{min}}$ and a max value $q_{\texttt{max}}$ of your choosing.
   - Use `GCV` or any flavor of vanilla `CV` you like.
   - Implement and use the `Nested Cross-Validation` scheme described at page 15 of Bates et al. (2022). Briefly explain what's the issue this method try to solve.
3. Finally graphically and numerically compare the two fits on the training and also on the test data. Which one do you prefer? Explain.